Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Màster en Enginyeria Informàtica

Feature selection with iterative feature weighting methods

Author:
Eng. Pol Rodoreda

Supervisor:
Dr. Lluís Belanche

Co-Supervisor:
Dr. Javier Larrosa

January 17, 2018

Contents

List of Figures

List of Tables

Abstract

In real-world concept learning problems, the representation of data often uses many features, only a few of which may be related to the target concept.

Determining which predictors should be included in a model is becoming one of the most critical questions as data are becoming increasingly high-dimensional. Feature selection has become very important in some areas that use datasets with hundreds or thousands of variables. These areas include text processing, gene expression array analysis, and combinatorial chemistry among others.

In this work we present a new way of feature selection. Our propose is based on the introduction of the Relief algorithm to approximate the optimal degree of influence of individual features to find which attributes are less important and have an unfavourable impact on the prediction model results. This will help us to remove worse attributes, that ones that are irrelevant in the dataset, and using an inducers determine the quality of the subset.

Using an experimental analysis we could see that the most complex model does not perform better results than the previous one. So, we can say that it's not useful to perform a classification every time we remove an attribute and it's only necessary to perform an importance classification at the beginning of the algorithm to achieve the best results.

Acknowledgements

This work would not have been possible without the guidance and contribution of my advisor, Dr. Lluís Belanche (Dept. de Ciències de la Computació, Universitat Politècnica de Catalunya, Barcelona). Thank you for the opportunity to acquire more knowledge in the machine learning world.

I want to say thank you to all my family (included you) for the support they have given to me during the course of these final master's project.

1 Introduction

The *feature selection* problem is ubiquitous in machine learning (ML) [James u. a., 2014] and data mining and its importance is beyond doubt. The main benefit of a good process of feature selection is the improvement of the inductive learner in terms of speed and simplicity of the induced model.

The high dimensionally problems has increase and becomes an interesting challenge for machine learning researchers. ML gets difficult when there are many features and very few samples, because the algorithm will not be able to distinguish correctly the relevant data and the noise.

In this work we use Relief algorithm [Kira und Rendell, 1992b] as feature weighting method to develop a new feature selection algorithm for regression and two-class classification problems. An extensive experimental study with five different sizes of artificial datasets gives us controlled experimental conditions to describe the results and improve the algorithms.

1.1 Goals

Machine learning uses *Feature Selection* (FS) methods to be able to solve problems with large number of features and noise. As we will see in section 2, FS is the process of detecting relevant features and discard the irrelevant ones.

Feature Weighting (FW) is a technique used to approximate the optimal degree of influence of individual features using a set of data.

The idea of this work is to develop a new feature selection algorithm using wrapper method idea (section 2) but not for all subset of data; we obtain a classification of the attributes due to its importance in the dataset and sending the result to a learner that will calculate the performance of the subset. To create the next subset we will remove the less important attribute. To achieve this, the main goals we propose in this work are:

- Make a comparison of a new algorithms of feature selection, each one increasing the previous one, exposing the performance results. These algorithms start with a basic algorithm which performs a Relief classification and ends with an algorithm that fusion Relief with a learner that calculates the subset performance.
- The datasets will be artificial in the first group of experiments, so we need to create regression and classification problems. We use synthetic data to validate the models by checking their performance before using real data.
- We also want to demonstrate that a complex algorithm that uses feature weighting for each subset of data improves the performance of an algorithm that does an initial classification of the attributes and only checks its inducers performance in each subset of data.

1.2 Motivation and related work

See how an algorithm can classify some data, or predict some results aroused in me the curiosity to know more about machine learning, so I took part in this project with the goal to improve my knowledge about this kind of technology, developing a useful work to achieve the main proposed goals.

Previous experimental work on feature selection include [Doak, 1992] and [Aha und Bankert, 1996]. This studies use artificial datasets such as *Parity*, *Led* or *Monks* problems. Demonstrate improvement on synthetic datasets can be more convincing than doing in typical real problems where the real solution is unknown the real solution.

Related with Relief, [Kira und Rendell, 1992b] introduces the algorithm making some test with artificial data to show the advantages in terms of learning time and accuracy of the learned concept. Few years later, [Kononenko, 1994] presents ReliefF which could handle problems with noisy, incomplete, and multi-class data sets. [Robnik-Sikonja und Kononenko, 1997] develop a new Relief algorithm called RReliefF that allows working with regression problems. Finally, in [Robnik-Sikonja und Kononenko, 2003] we can find an analysis of ReliefF and RReliefF with some modifications in the main code in order to improve the results.

1.3 Project Structure

This project is organized as follows. We first overview Feature Selection and Feature Weighting in section 2, present both and explain how this methods work. We also describe an introduction to Relief algorithm. A description of models that have been used to develop the work could be read in section 3. In section 4 we present the creation of an artificial dataset for regression and classification problems to develop the models and a description of real datasets. To see the performance of the study, section 5 describes the experimental part of the work. Finally, we conclude the study in section 6.

2 An overview of Feature Selection and Feature Weighting

Feature selection (FS) is also called variable selection or attribute selection. Is the process of detecting the relevant features and discarding the irrelevant ones, with the goal of obtaining a subset of features that can give relatively the same performance without significant degradation. Fewer attributes is desirable because it reduces the complexity of the model, and a simple model easier to understand and explain.

FS is different from dimensionally reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionally reduction method do so by creating new combinations of attributes, where feature selection exclude and include attributes presents in dataset without changing them. The central premise when using a feature selection technique is that the data contains many features that are either *redundant* or *irrelevant*, and can be removed without incurring much loss of information. *Redundant* or *irrelevant* features are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated.

We also need to differentiate feature selection from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of features.

The feature selection methods are typically presented in three classes based on how they combine the selection algorithm and the building model:

Filter methods

Filter feature selection methods (figure 1) select variables regardless of the model. They are based only on general features like the correlation with the variable to predict. These methods suppress the least interesting variables and other variables will be part of a classification or regression model.

However, filter methods tend to select redundant variables because they do not consider the relationships between variables.

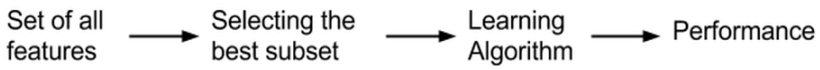

Figure 1: Filter method for feature selection

Wrapper methods

This kind of feature selection methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model is used to evaluate a combination of features and assign a score based on model accuracy (figure 2).

The main disadvantages of these methods are:

- The increasing overfitting when the number of observations is insufficient.
- The computation time when the number of variables is large.

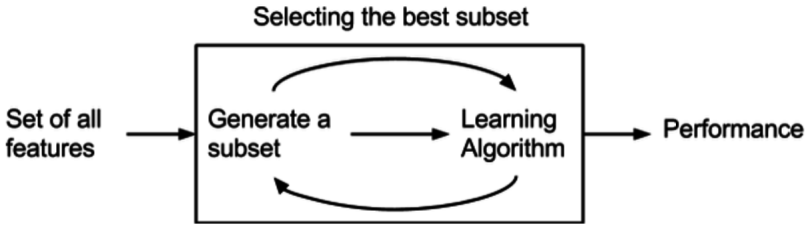

Figure 2: Wrapper method for feature selection

Embedded methods

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. A learning algorithm takes advantage of its own variable selection process and performs feature selection and classification simultaneously (figure 3). An example of embedded method for feature selection would be a decision tree.

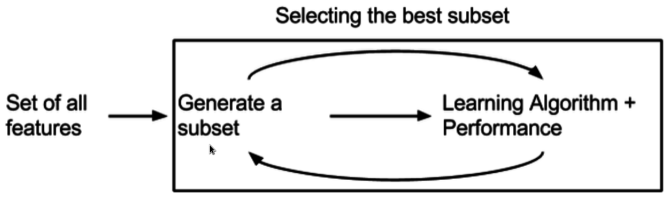

Figure 3: Embedded method for feature selection

Feature weighting (FW) is a technique used to approximate the optimal degree of influence of individual features using a training dataset. When successfully applied relevant features are attributed a high weight value, whereas irrelevant features are given a weight value close to zero. FW can be used not only to improve classification accuracy but also to discard features with weight below a certain threshold value.

2.1 Relief family of algorithms

In this section we present an introduction into the Relief family of algorithms. First we present the original Relief algorithm [Kira und Rendell, 1992b] which was limited to classification problems with two classes. We discuss its extension ReliefF [Kononenko, 1994] which can deal with multiclass problems, is more robust and can deal with incomplete and noisy data. Finally, we describe how ReliefF was adapted to develop RReliefF [Robnik-Sikonja und Kononenko, 1997] which works in regression environments.

2.1.1 Relief

A key idea of the original Relief algorithm, given in figure 4, is to estimate the quality of attributes according to how well their values distinguish between instances that are near to each other. For that propose, given a randomly selected instance R_i, Relief searches for its two nearest neighbours: one from the same class, called nearest hit NH_X, and the other from the different class, called nearest miss NM_X. It updates the quality estimation W[A] for all attributes A depending on their values for R_i, NH_X and NM_X.

It is based on a simple principle: we like to put objects with similar properties in a class. Some of these properties (or features) are very important in the classification task and others are less important.

Relief is considered one of the most successful algorithms for assessing the quality of features due to it's simplicity and effectiveness. It is a feature weighting based algorithm inspired by instance-based learning that consists in three important parts:

1. Calculate the nearest miss and nearest hit.
2. Update the weight of a feature.
3. Return a ranked list of features or the top k features according to a given threshold.

The algorithm starts initializing the weight vector and setting the weight for every feature to 0. Then, it picks a random feature X and calculates the nearest hit (NH_X) and the nearest miss (NM_X). In case of numerical data, the distance to use is *Euclidean distance*.

Algorithm 1 Relief

Input: for each training instance a vector of attribute values and the class value
Output: the vector W of estimations of the qualities of attributes

```
1: set all weights W[A] := 0.0;
2: for i := 1 to m do
3:    randomly select an instance R_i;
4:    find nearest hit NH_X and nearest miss NM_X;
5:    for A := 1 to a do
6:       W[A] := W[A] - diff(A, R_i, NH_X)/m + diff(A, R_i, NM_X)/m;
7:    end for
8: end for
```

Figure 4: Relief algorithm

Function diff(A, I_1, I_2) calculates the difference between the values of the attribute A for two instances I_1 and I_2. For nominal attributes it was originally defined as:

$$diff(A, I_1, I_2) = \begin{cases} 0 & value(A, I_1) = value(A, I_2) \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

and for numerical attributes as:

$$diff(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)} \tag{2}$$

As we mentioned, original Relief cannot deal with incomplete data and is limited in two-class problems. Its extension, which solves these and other problems, is called ReliefF.

2.1.2 ReliefF

This extension of the original Relief algorithm (figure 5) works similar but after select randomly an instance R_i, it searches for k of its nearest neighbours from the same class, called nearest hits H_j, and also k nearest neighbors from each of the different classes, called nearest misses $M_j(C)$; the update formula of weights vector is similar to that of Relief, except that it averages the contribution of all the hits and all the misses. This gives ReliefF greater robustness concerning noise and incomplete data.

Algorithm 2 ReliefF

Input: for each training instance a vector of attribute values and the class value
Output: the vector W of estimations of the qualities of attributes

1: set all weights W[A] := 0.0;
2: **for** i := 1 to m **do**
3: randomly select an instance R_i;
4: find k nearest hits H_j;
5: **for** each class C $\neq$ class(R_i) **do**
6: from class C find k nearest misses M_j(C);
7: **for** A := 1 to a **do**
8: $$W[A] := W[A] - \sum_{j=1}^{k} diff(A, R_i, H_j)/(m \cdot k) + \sum_{C \neq class(R_i)} \left[\frac{P(C)}{1-P(class(R_i))} \sum_{j=1}^{k} diff(A, R_i, M_j(C)) \right]/(m \cdot k);$$
9: **end for**
10: **end for**
11: **end for**

Figure 5: ReliefF algorithm

2.1.3 RReliefF

We finish the description of the Relief family of algorithms with RReliefF (figure 6). Relief's estimate W[A] of the quality of attribute A is an approximation of the following difference of probabilities:

$$\begin{aligned} W[A] = P(\text{diff. value of A}|\text{nearest inst. from diff. class}) - \\ P(\text{diff. value of A}|\text{nearest inst. from same class}) \end{aligned} \tag{3}$$

It deal with regression problems, where the predicted value $\tau(\cdot)$ is continuous; therefore nearest hits and misses cannot be used. To solve this issue, instead of search if two instances belong to the same class or not, a kind of probability that predicted values of two instances are different is introduced.

Algorithm 3 RReliefF

Input: for each training instance a vector of attribute values $\mathbf{x}$ and the predicted value $\tau(\mathbf{x})$

Output: the vector W of estimations of the qualities of attributes

set all N_{dC}, $N_{dA}[A]$, $N_{dC\&dA}[A]$, W[A] to 0;
for i := 1 to m **do**
  randomly select an instance R_i;
  select k instances I_j nearest to R_i;
  for j := 1 to k **do**
    $N_{dC} := N_{dC} + \text{diff}(\tau(\cdot), R_i, I_j) \cdot d(i, j)$;
    for A := 1 to a **do**
      $N_{dA}[A] := N_{dA}[A] + \text{diff}(A, R_i, I_j) \cdot d(i, j)$;
      $N_{dC\&dA}[A] := N_{dC\&dA}[A] + \text{diff}(\tau(\cdot), R_i, I_j) \cdot \text{diff}(A, R_i, I_j) \cdot d(i, j)$;
    end for
  end for
end for

Figure 6: RReliefF algorithm

2.2 Inducers

In this section we describe the inducers we will use to obtain the quality of the classification result each time we remove an attribute.

2.2.1 Linear regression

Is a linear approach for modelling the relationship between a dependent variable y and one or more independent (or explanatory) variables denoted X. The case of one explanatory variable is called simple regression. In this case we use **multiple linear regression** because we have more than one independent variable.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of y given the value of X is assumed to be an affine function of X; less commonly the median or some other quantile of the conditional distribution of y given X is expressed as a linear function of X.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.
- Given a variable y and a number of variables X_1, ..., X_p that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship

between y and the X_j, to assess which X_j may have no relationship with y at all, and to identify which subsets of the X_j contain redundant information about y.

To perform a linear regression problem in R we use caret package ([Kuhn, 2017]) to perform a linear regression with 10-fold cross validation to obtain the R-squared value to determine how well the model fits the data.

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or coefficient of multiple determination for multiple regression.

The definition of R^2 is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or:

- R-squared = Explained variation / Total variation
- R-squared is always between 0 and 100%:
  - 0% indicated that the model explains none of the variability of the response data around its mean.
  - 100% indicates that the model explains all the variability of the response data around its mean.

2.2.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

In this case we use R package MASS ([Ripley, 2017]) that gives us **lda** function that returns the prior probability of each class, the counts of each class in the data, and other parameters related with the model.

To obtain the accuracy of the model, we execute **lda** with argument `CV = TRUE` that performs a leave-one-out cross validation predictions of the class; then we compare the result with the original values and calculate the sum of the percent correct predictions to calculate the accuracy of the model.

2.2.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' tendency of overfitting to their training set.

In this case we use R `randomForest` package ([Liaw und Wiener., 2015]). This package gives us **randomForest** function that uses, by default, 500 trees to construct the model. When we work in a regression context, the result will give us the R-squared for each decision tree and we can perform the mean to obtain the general R-squared.

In classification we abstract the OOB error, also called out-of-bag estimate. It is method of measuring the prediction error of random forests and other machine learning models utilizing bootstrap aggregating to sub-sample data samples used for training. OOB is the mean prediction error on each training sample x_i, using only trees that did not have x_i in their bootstrap sample.

3 Description of proposed models

As we explain previously, the main goal of this work is to explore the combination of Relief family algorithms with an inductor in order to create synergies between them for their interaction.

In order to carry out a gradual development, our work have been structured in 3 steps or models. Each model improve the previous one adding some new features in order to see how is the effect.

3.1 Model 0

This model is the starter point of the work. It uses Relief as feature weighting algorithm to obtain the classification of the attributes. We only use regression because our goal is to learn how Relief works.

We develop a cutting function in order to find which is the best point i^* to cut the ranked dataset given a vector of importance values (w). The first step is to convert all the classification values to positive using equation 4, sort from highest to lowest and search the *argmax* as we can see in equation 5. The output of the model is the vector of best attributes and the linear regression performance with all attributes and after cut the data set.

$$if \quad min(w) < 0 \quad \text{then} \quad w_i = w_i - min(w), i = 1, ..., a \tag{4}$$

$$i^* := argmax_{i=2,...,a} = \frac{\overline{w}_i}{\overline{w}_{i-1}} \quad where \quad \overline{w}_i = \frac{1}{i} \cdot \sum_{k=1}^{i} w_k \tag{5}$$

Then, we use this vector to remove worst attributes from the original dataset to obtain a smaller one with same performance as the original.

3.2 Model 1

Model 0 is simple, but necessary; we create model 1 that, like the previous, uses Relief to obtain a vector of importance values (w) at the beginning.

This time we use an inducers, depending on the problem type (linear regression, linear discriminant analysis or random forest) to evaluate the performance of each subset removing the worst attribute at each iteration as we can see in figure 7.

The result is a plot of the inducers's performance for each step, how and which attributes we need to obtain the best result and, if the data is synthetic, Kendall's correlation coefficient (section 3.4).

Figure 7: Model 1 flowchart

3.3 Model 2

To improve the previous model we create model 2 that, at each iteration recalculates the classification using Relief as we can see in figure 8. We do it because when the worst attribute is removed, the relationship between the other attributes can change.

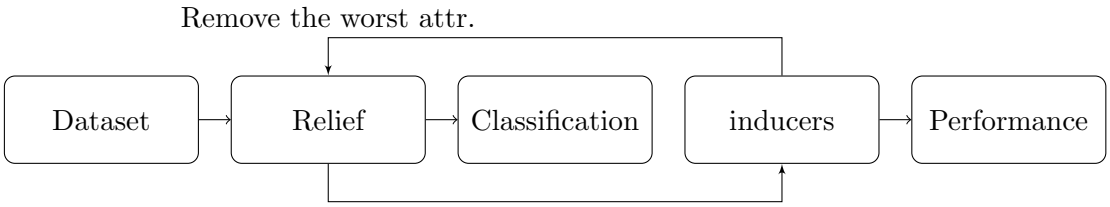

Figure 8: Model 2 flowchart

Like model 1 the result is a plot of the inducers's performance, the attributes we need for best result and Kendall's correlation coefficient if data is artificial.

3.4 Kendall Rank Coeficient

To evaluate if the result we obtain is confident in model 1 and model 2, we use a truth vector with the rank of the variables in the dataset and we compare it with the result vector using Kendall rank coefficient. This coefficient can be only calculated if dataset is artificial because we can know the solution of the problem.

The correlation coefficient is a measurement of association between two random variables. While its numerical calculation is straightforward, it is not readily applicable for non-parametric statistics. A more robust approach is to compare the rank orders between the variables.

Next, we present an example of how we treat the truth and solution vectors in order to calculate Kendall's correlation coefficient. First we assign the known classification to truth vector depends on the attribute quality. Then we assign attributes names to the corresponding position:

Attr. 1	Attr. 2	Attr.3	Attr. 4	Attr.5	Attr. 6	Irrel. Attr. 7	Irrel. Attr. 8
1	2	3	5	5	6	10	10

Once we have the truth vector we execute the algorithm and we store which attribute is removed at each iteration in a variable. Reversing the result will be used to construct the names of the solution vector which at its initialization has the same classification values as truth vector:

Attr. 2	Attr. 1	Attr.4	Irrel. Attr. 7	Attr.3	Attr. 6	Attr. 5	Irrel. Attr. 8
1	2	3	5	5	6	10	10

To be able to check if the algorithm performs a good classification we need to sort the solution data using names of the truth vector, so we will obtain:

Attr. 1	Attr. 2	Attr.3	Attr. 4	Attr.5	Attr. 6	Irrel. Attr. 7	Irrel. Attr. 8
2	1	5	3	10	6	5	10

To find the Kendall coefficient between truth vector and solution vector, we create a matrix m consisting of the values of two vectors. Then, we apply the function *cor* with option "kendall". The result will be a number between 0 and 1 where a value of 1 indicates maximum correlation.

4 Datasets

Data is a set of values of qualitative or quantitative variables. Pieces of data are individual pieces of information. While the concept of data is commonly associated with scientific research, there are range of organizations and institutions including businesses, governments and non-governments organizations, that collects data.

Data can be measured, collected, analysed and visualized using graphs or images. Raw data (or *unprocessed data*) is a collection of numbers or characters before it has been "cleaned" and corrected by researchers. This kind of data has to be corrected to remove outliers and instruments or data entry errors. Experimental data is data that is generated within the context of a scientific investigation. Artificial data is data that is generated using a known formula, so it allows the researcher to know the real solution of the problem.

This work uses two types of datasets: artificial or synthetic and real data. In first case we prepare data for regression and classification problems using different sizes: 100, 200, 400, 800 and 1600 instances. This will be used later in experimental section to study the performance of the algorithm.

The following lines explain how artificial datasets are created and which kind of real data we use.

4.1 Artificial datasets

In feature selection it is not possible to know if the solution of your model is correct, so to evaluate model's performance we use artificial data.

For regression we use Friedman 1 benchmark problem from mlbench R package (Leisch und Dimitriadou [2010]). In the original formula, inputs are 10 independent variables uniformly distributed on the interval [0, 1] but only 5 are actually used. The output is generated according to the formula

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e \tag{6}$$

where e is N(0, sd) and *standard deviation* $= 1$ by default.

So, we depart from Friedman 1 to generate 5 relevant and 5 irrelevant attributes. Then we add 17 new irrelevant variables using a Gaussian normal distribution with mean and standard deviation equals 0 and 1 respectively. Finally we generate three redundant attributes, two calculating the product between first and second relevant variables plus some noise and the third redundant is the mean of relevant variables three, four and five.

For two class classification problems we also use Friedman 1 benchmark with default values and 10 attributes, converting Target values using next criteria:

$$Target = \begin{cases} True & \text{if } value > 14.4 \\ False & \text{otherwise} \end{cases} \tag{7}$$

Rel.1	Rel.2	...	Irrel.U.1	Irrel.U.2	...	Red.1	Red.2	...	Target
0.2428	0.9243		-1.4828	-1.7055		0.3249	-0.5193		13.6295
-1.7267	1.0562		0.3108	-1.1856		2.0010	1.1282		11.9445
-0.7543	-0.0693		-0.9458	-0.7721		-1.2596	0.1096		19.3056
-0.8102	-0.3795		-1.406	-0.7150		0.4428	1.2828		16.0829

Table 1: A piece of regression artificial dataset

We choose a value of 14.4 because it allow us to obtain a two-class problem with balanced size for classification and we can use the real Relief algorithm.

4.2 Real datasets

To test the algorithm with real data, we select some datasets from the UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml/index.php`).

For regression problem we use:

SkillCraft1 [Mark B. und C., 2013]

These data was collected from Real-Time Strategy game players from 7 levels of expertise, ranging from novice to full-time professionals.

It is a collection of data from 3395 unique games played. The data was extracted from recorded game replays. Every observation in the dataset is identified with a unique and non-missing Game ID. The data consists of 20 variables including Game ID. These variables pertain to various for each game such as Basic Descriptive data, on screen movement, shortcut assignment and unit related. Of the 20 observations 15 were of continuous nature and were used as the variables used to do the analysis. The variable being predicted is League Index, which is ordinal with values 1-8.

Parkinson Telemonitoring [T. und L., 2009]

This dataset is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The recordings were automatically captured in the patient's homes. Each row corresponds to one of 5,875 voice recording from these individuals.

The main aim of the data is to predict the motor and total UPDRS scores ('motor_UPDRS' and 'total_UPDRS') from the 16 voice measures. In this work we only use total_UPDRS.

For classification problems, we use:

Breast Cancer Wisconsin [Wolberg, 1995]

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. In this work we use the first dataset that has 369 instances and 10 attributes plus the target

value. The main goal is to determine if a cancer is malignant or benign (2 or 4).

Ionosphere [Sigillito, 1989]

This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. It consist in a dataset of 351 instances and the targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere.

Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

5 Experimental evaluation

After describing the models we have developed and what kind of datasets we will use in our experiments, in the next subsections we will show the results we have obtained.

To develop this work, we use R language ([Team, 2008]) and its `IDE R Studio`. `R` is a free, open-source software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows ans MacOS and it's supported by the R Foundation for Statistical Computing. This language offers a lot of packages related with statistics, machine learning, data mining, etc.

We use CORElearn package ([Marko Robnik-Sikonja, 2017]) and its routine *attr.eval* with estimator *RReliefFequalK* or *Relief* to perform Relief algorithm in regression and classification problems.

As we describe in section 2.2 we use `R` packages for linear regression in regression problems, linear discriminant analysis in classification problems and random forest for both types.

All the results can be found in appendix section.

5.1 Artificial data

We start our experimental analysis using artificial data to check the performance of model 0; as we can see in section 3.1 this model returns us the linear regression performance after cut the dataset. We store and present this result into table 2 to compare later with other models.

5.1.1 Model 0

As we mentioned in model explanation section, here we only use regression data, so we show results for all dataset sizes. In order to see the performance of the model, we use linear regression and validate the result using 10-fold cross validation. We compare the result obtained at the beginning with all attributes with the result obtained after remove attributes with worst position in Relief classification.

	Dataset size				
	100	200	400	800	1600
All attributes	0.5703685	0.6921257	0.7293787	0.7410310	0.7509261
After cut	0.5258170	0.6165967	0.7441911	0.7368039	0.1628267

Table 2: Model 0 cross validation results for regression problem

We can see that model 0 increases it's performance if dataset has more instances, but only overcome the initial result when it has 400 rows of data. We also want to note that the worst result happens with 1600 instances, because the algorithm only returns *Rel. 2* attribute, so the linear regression after cut is poor.

For a better look to the results we plot it in figure 9.

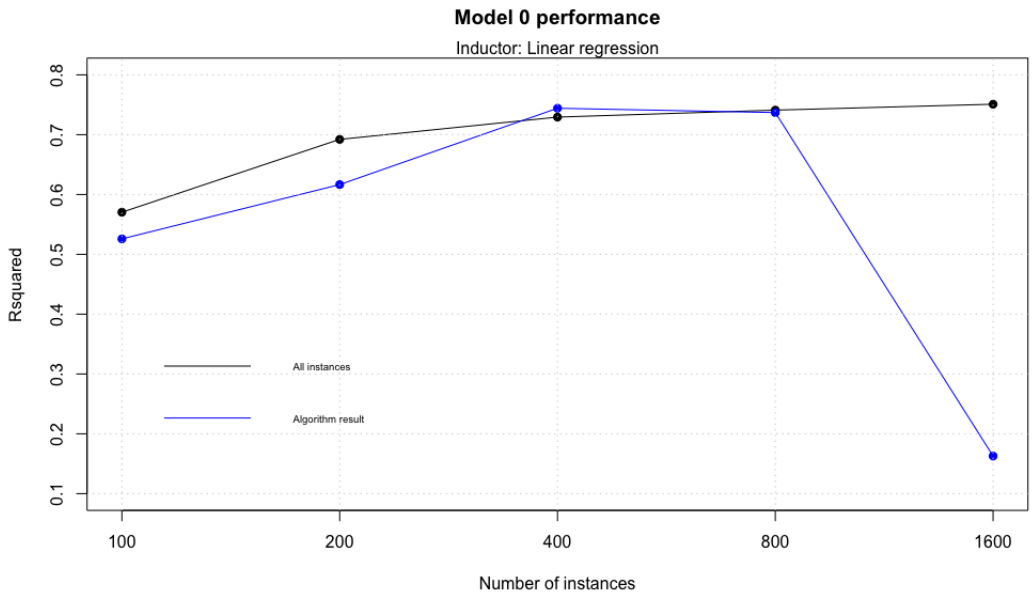

Figure 9: Model 0 cross validation results

5.1.2 Model 1

We have seen that model 0 is too simple and it does not present a good performance when we apply it to our dataset. In model 1 we first work with artificial regression problems to develop the base of the algorithm. In the next lines we show and describe the results of the algorithm.

For a better understanding we use plots where each point indicates which attribute is deleted before calculate the inducers's performance. In first attempts we use linear regression as inducers and calculate 10-fold cross validation to obtain the R-squared (R^2) value at each iteration. There are two basic parameters we want to check after algorithm execution; first is the best inducers's result and second with how many attributes it is achieved.

We start with a dataset of 100 instances; in figure 10 we can see that the algorithm is irregular and the Relief classification is bad since important variables are removed in the first iterations. As we can see in table 3, the algorithm needs 27 of 31 attributes to get the best R^2 result.

The performance increase when we use a dataset with 400 instances; the algorithm remove most of the irrelevant variables first and has a decrease on R-squared value when removes *Rel. 5* attribute, but it isn't good as we expect because in some final iteration it removes *Irrel. N. 5* attribute and the performance decreases a lot.

If we compare now the two last sizes, 800 and 1600 instances, we can see that the model performance is better than others, but not much. In this case it's more interesting to take a look to the number of attributes the algorithm needs to obtain the best performance;

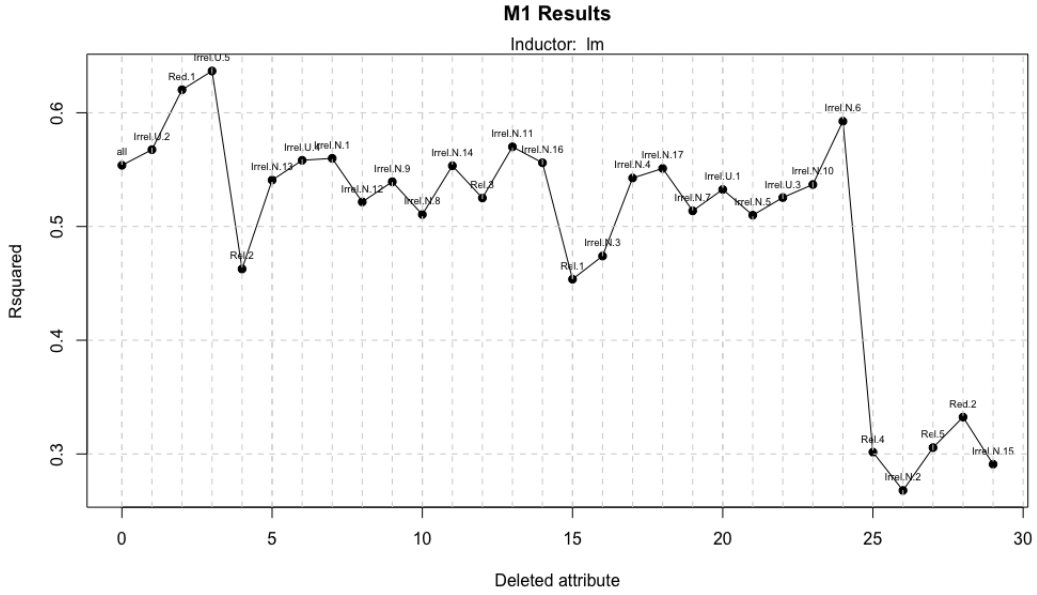

Figure 10: Model 1 execution with a dataset of 100 instances

in table 3 we can see that model 1 obtains its best performance with 1600 instances but it needs 12 of 31 attributes. If we take a look to 800 instances, the performance is a little bit smaller but it only needs 5 attributes.

	Number of instances				
	100	200	400	800	1600
lm	27/31 0.63673	27/31 0.69635	19/31 0.74345	5/31 0.74678	12/31 **0.75485**
rf	27/31 0.53445	27/31 0.63635	13/31 0.74399	6/31 0.84605	6/31 **0.87964**

Table 3: Number of attributes model 1 needs to best performance in regression context

Using Kendall's correlation coefficient results (table 4) we can validate that in model 1 the best performance is achieved when the algorithm works with an 800 instances dataset. This values are similar if we use random forest because this model performs Relief classification once at the beginning.

Number of instances				
100	200	400	800	1600
0.2587065	0.3781095	0.4029851	**0.7910448**	0.6915423

Table 4: Kendall's correlation coefficient results for model 1 using regression dataset

When using random forest as inducers, we can appreciate a better performance when we use bigger datasets (table 3). This time we securely validate that with a dataset of 1600 instances we obtain the best performance result of model 1 in regression since it uses 6 attributes to obtain an R^2 of 0.87964. Figure 11 show the execution of this best performance.

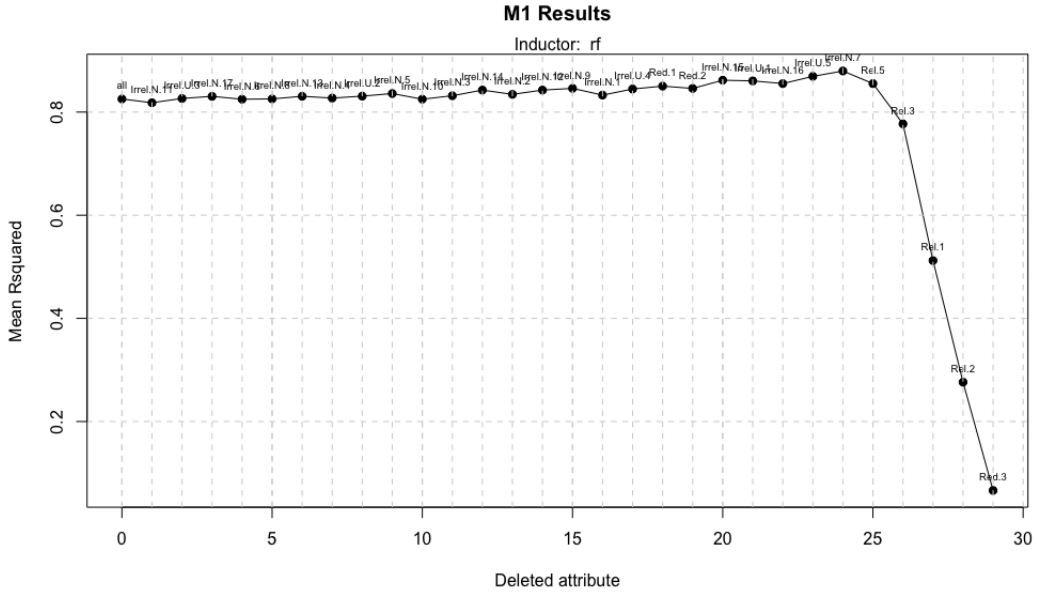

Figure 11: Model 1 execution with a dataset of 1600 instances

If we now take part in classification problems we can see that using both inducers, Linear Discriminant Analysis and Random Forest, the results improve while dataset size increase. As we can see in table 5, with LDA we achieve the best performance using a dataset with 800 instances but it needs 9 of 11 attributes; with 400 instances (figure 12), the accuracy is a little bit lower but it only uses 4 attributes.

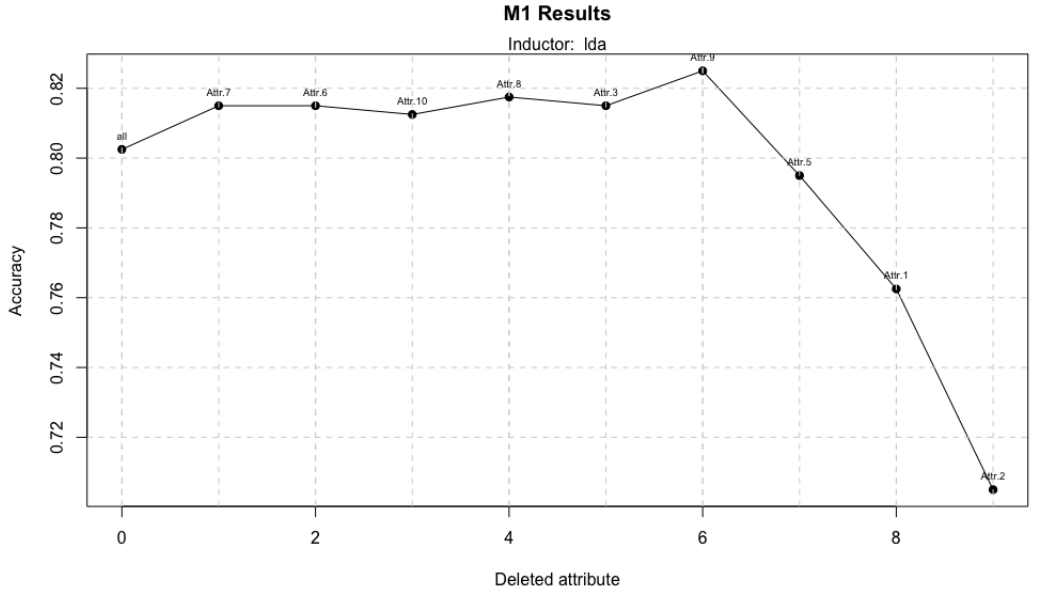

Figure 12: Model 1 performance using LDA and a dataset with 400 attributes

Let's look the results if we use Random Forest. In this case the better performance is achieved with the largest dataset and it uses half of the attributes; so it is a nice

result to see. In figure 13 we can see the performance that outputs the best result in the classification context.

	Number of instances				
	100	200	400	800	1600
lm	10/11 0.79	10/11 0.79	4/11 0.825	9/11 **0.8462**	10/11 0.8437
rf	3/11 0.8244	5/11 0.8291	6/11 0.8475	7/11 0.86735	5/11 **0.88288**

Table 5: Number of attributes model 1 needs to best performance in classification context

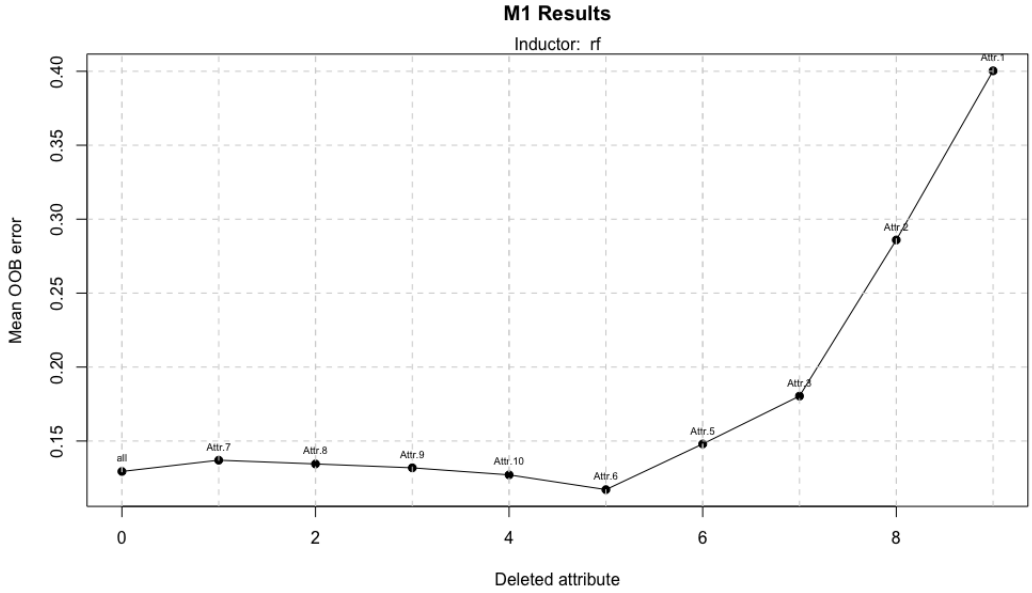

Figure 13: Model 1 performance using RF and a dataset with 1600 attributes

Kendall's correlation coefficient confirm that in classification, the algorithm has a better performance if number of instances increase as we can see in table 6. It helps us to determine that in a classification context, use a Random Forest as inducers is the best choice.

Number of instances				
100	200	400	800	1600
0.4117647	0.3529412	0.6176471	0.7058824	**0.8529412**

Table 6: Kendall's correlation coefficient results for model 1 using classification dataset

5.1.3 Model 2

After an extension review of model 1, the goal of experimenting with model 2 is to see that, performing a Relief classification every time we remove the worst attribute, it increases the algorithm quality.

Like in model 1 we begin with regression artificial problems. In table 7 we can appreciate that, another time, when the number of instances increase the result of the inducers also increases, but this is natural for the majority of the algorithms. The performance is similar, even less to model 1 in some cases so we have to better look the number of attributes the model needs to achieve the best performance to take part in a final decision about which is the best case.

If we use Multi Linear Regression the number of attributes is lower than model one, except if we use a dataset with 800 instances where the value in model 2 is not common. With Random Forest we can appreciate a little difference if we compare to model 1 because it needs less attributes in some cases but not when we use the biggest dataset, where we obtain the best performance value.

	Number of instances				
	100	200	400	800	1600
lm	27/31 0.6384	19/31 0.72801	12/31 0.74236	21/31 0.74778	5/31 **0.75349**
rf	27/31 0.51601	12/31 0.67496	9/31 0.75721	6/31 0.83847	6/31 **0.87626**

Table 7: Number of attributes model 2 needs to best performance in regression context

In this case, we use Kendall's correlation coefficient represented in table 8 to evaluate which inducers we use to validate model's 2 performance. We can see that, although the inducers results are similar in both cases, the best score is achieved when we use Random Forest and the largest dataset and this result matches with the best R^2 value obtained. We can see the best performance in figure 14, where we can see that in the final iterations the inducers result decreases a lot because the algorithm deletes the most important attributes like *Rel. 3* or *Rel. 1*.

	Number of instances				
	100	200	400	800	1600
lm	0.1691542	0.4129353	**0.800995**	0.5671642	0.7064677
rf	0.1691542	0.4129353	0.800995	0.5671642	**0.8208955**

Table 8: Kendall's correlation coefficient results for model 2 using regression dataset

Now its time to send classification data to model 2. Linear Discriminant Analysis inducers presents the same performance as we saw in model 1, so next we explain the results obtained using Random Forest.

The improvement of the accuracy using Random Forest as inducers is small, but reaches the maximum value when we use the biggest dataset. It takes 6 of 9 attributes, while with a dataset of 200 instances it only needs 4 attributes to perform a very similar result, but the difference in the result is minimum so, another time, we need Kendall's correlation coefficient.

If we want to use Kendall's correlation coefficient to determine which is the best case in model 2 using classification data, we find that LDA and RF present the same value, inclusive if dataset has 400, 800 or 1600 instances. So, in this case we can determine that in a classification context with a Ranfom Forest, the algorithm works better when the

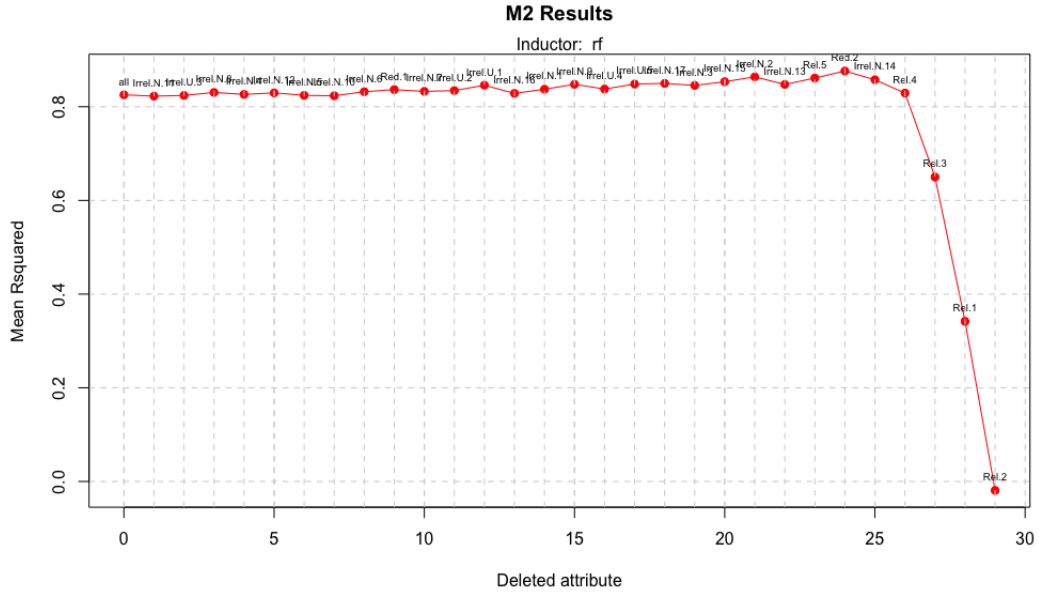

Figure 14: Model 2 performance using RF and a dataset with 1600 attributes

	Number of instances				
	100	200	400	800	1600
lm	10/11 0.79	10/11 0.79	4/11 0.825	9/11 **0.8462**	10/11 0.8437
rf	5/11 0.81432	4/11 0.84123	5/11 0.85598	9/11 0.87038	6/11 **0.88469**

Table 9: Number of attributes model 2 needs to best performance in classification context

dataset is bigger. We can see the result using a 400 instances dataset in figure 15.

	Number of instances				
	100	200	400	800	1600
lda and rf	0.6176471	0.4117647	**0.7941176**	**0.7941176**	**0.7941176**

Table 10: Kendall's correlation coefficient results for model 2 using classification dataset

5.1.4 Comparison between model 1 and model 2

After reviewing model 1 and model 2 results now its time to make a comparison to determine which model is better. In next lines we will compare parameters such as Kendall's correlation coefficient, inducers best performance and time execution.

All the experiments have been done using a MacBook Pro Mid 2012 with specifications:

- **Processor**: 2.5 GHz Intel Core i5
- **Memory**: 16 GB 1600 MHz DDR3

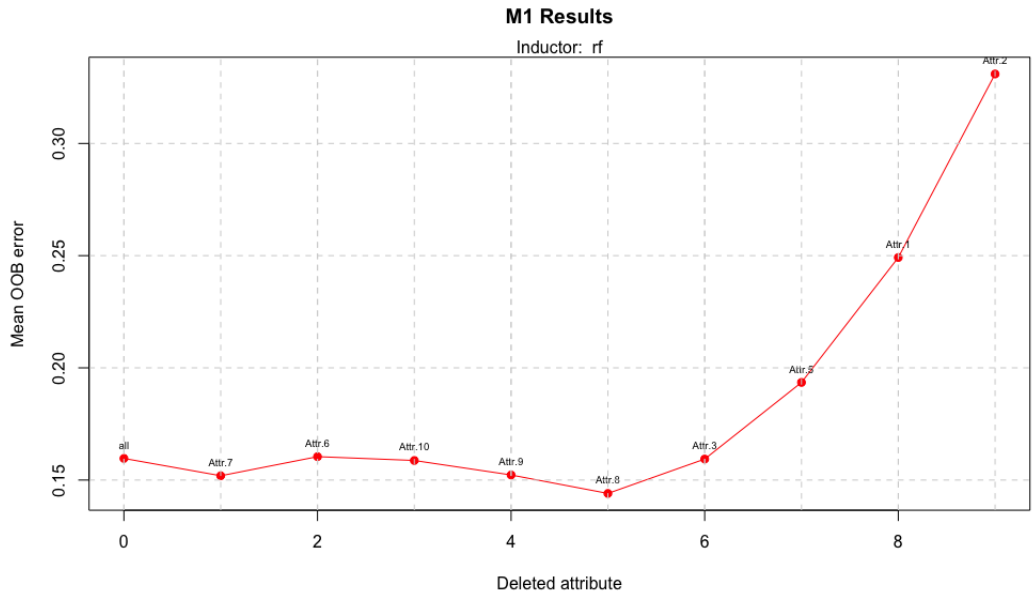

Figure 15: Performance of model 2 using a dataset with 400 instances and RF

- **Graphics**: Intel HD Graphics 4000 1536 MB
- **OS**: MacOS High Sierra 10.13.2

Let's start with regression data. In figure 16 we can appreciate that Random Forest present a better performance than Linear Regression and, for a minimum difference, the best result is achieved by model 1.

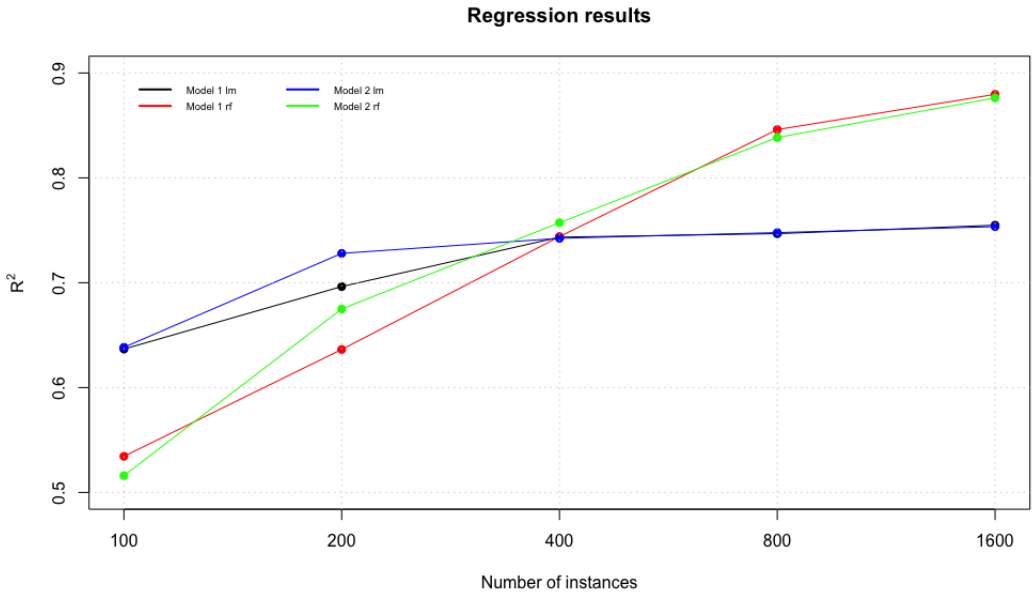

Figure 16: Regression results

But in this work we don't search for the best performance, but the best feature selection

algorithm; that means that we need the algorithm that removes the worst attributes first. This can be checked in figure 17 where we can see that when we use Random Forest in model 2 we obtain the best result.

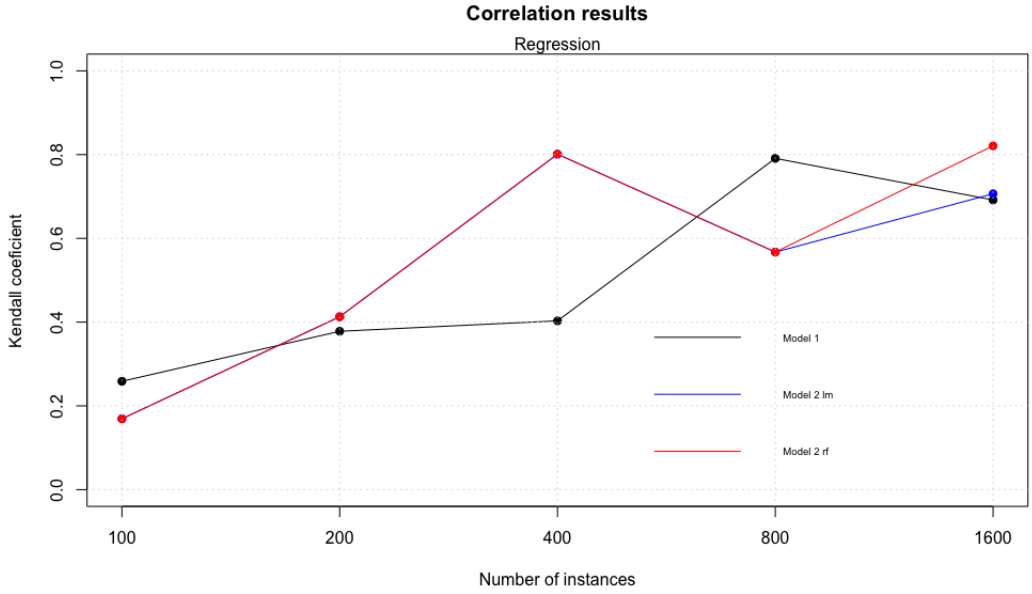

Figure 17: Kendall's correlation coefficient for regression results

In the experiments related with classification if we look to the accuracy results (figure 18) we can see that model 2 achieves the better performance using Random Forest but the difference with model 1 is very small. When the inducers in LDA we can't see a difference because both models perform the same result.

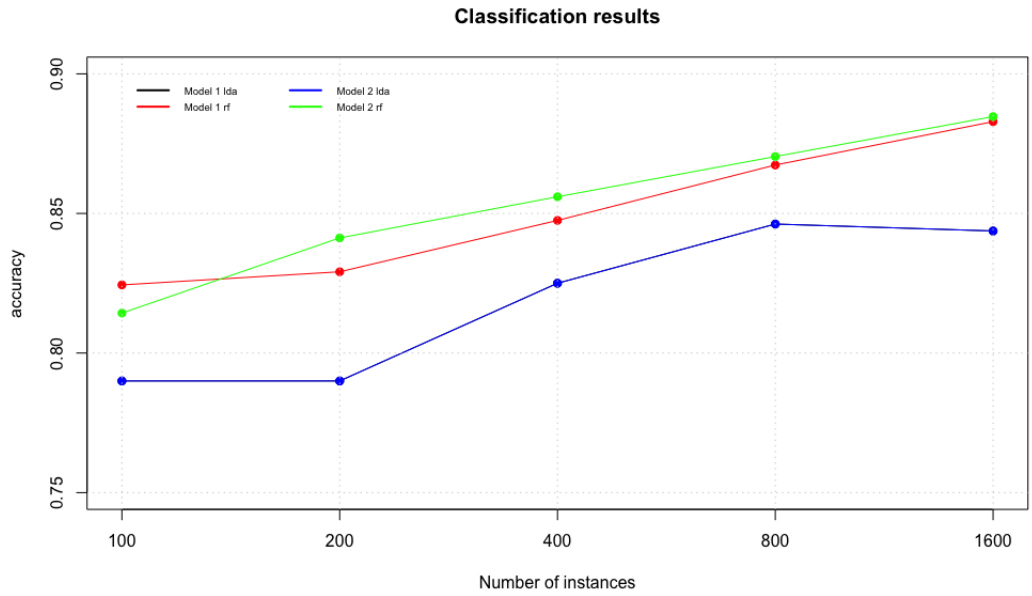

Figure 18: Classification results

It's time to evaluate Kendall's correlation coefficient (figure 19) to determine which is the best model in a classification context. Another time model 2 performs the better result when using Random Forest as inducers. But what happens if we look for execution time?

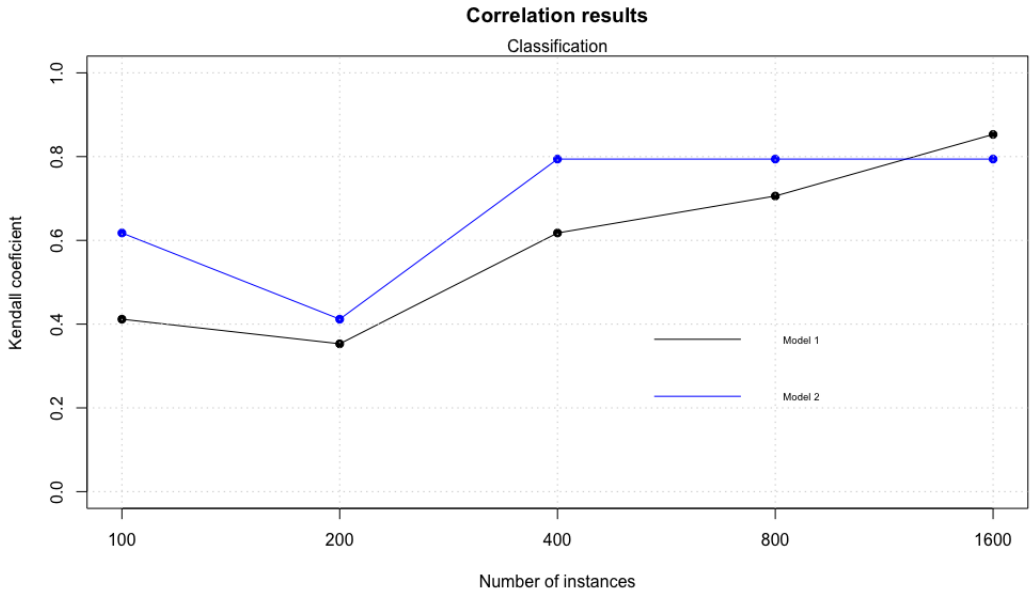

Figure 19: Kendall's correlation coefficient for classification results

In this work the times are relatively small, but figures 20 and 21 maybe can help us to predict a time execution with a bigger dataset.

We can see that in both cases Random Forest execution is slower than Linear Regression or Linear Discriminant Analysis, and all inducers present a time increment as we expected if dataset size increases.

In this point we have reviewed all models, in a regression and classification environment, with different dataset size and we also review Kendall's correlation coefficient to determine which is the best in every case. Now, its time to the user to decide which model wants to use and which inducers uses.

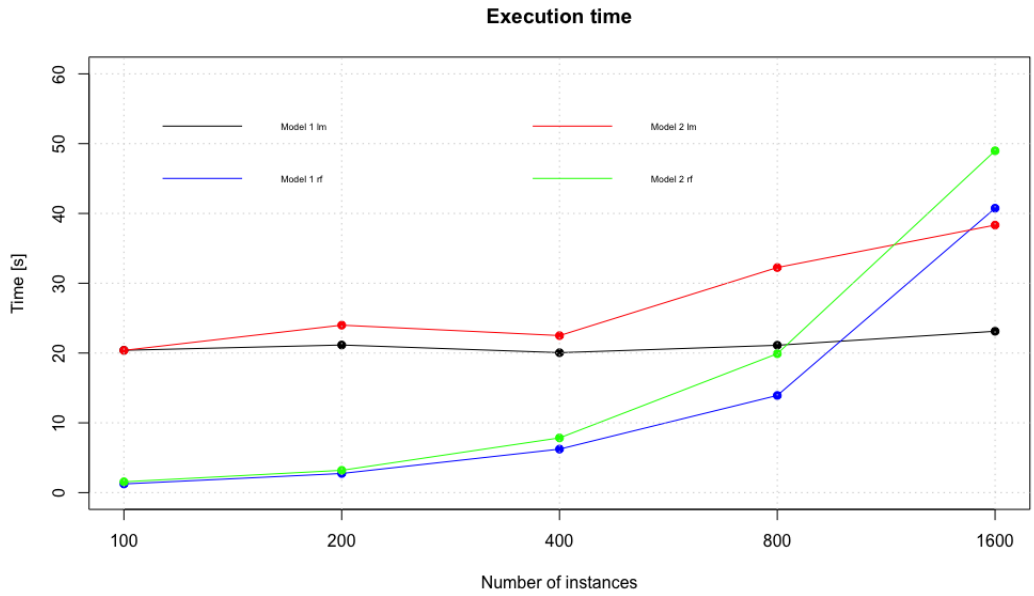

Figure 20: Execution time results using artificial regression data

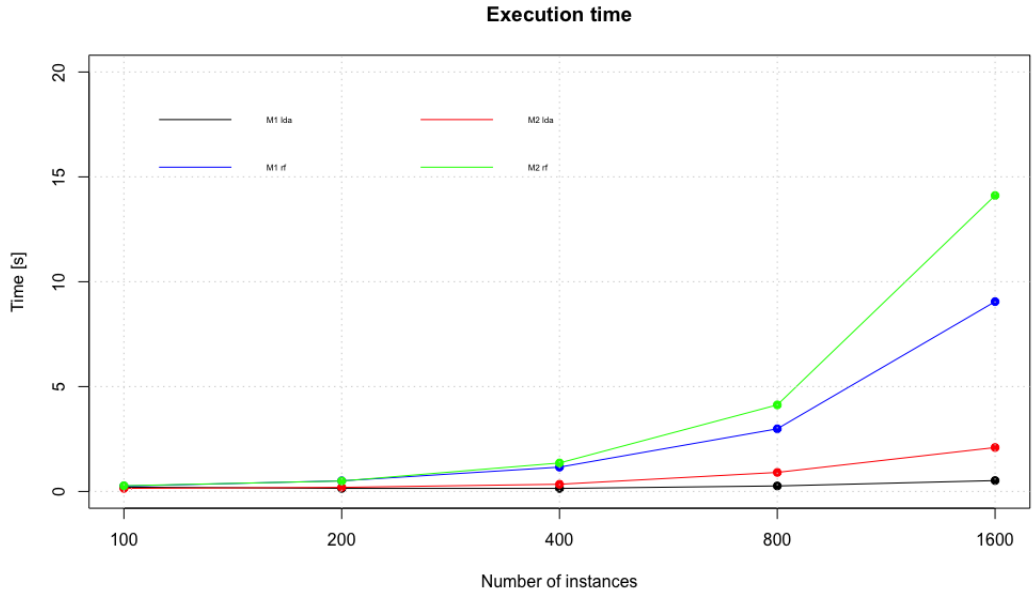

Figure 21: Execution time results using artificial classification data

5.2 Real data

After execute the algorithms in a controlled environment using synthetic data, its time to use real data. In this case we can't know how good the algorithm is because we don't know the real solution, we can only check the performance. For all real datasets we apply both models and represent in different tables the results we have obtained.

5.2.1 SkillCraft1

In terms of number of attributes, model 1 with Linean Regression (figure 22) present the best result as we can see in table 11. The difference between LM and RF in terms of performance is very small, so its not an important thing to take care of.

	Model 1	Model 2
lm	5/20 0.494997	19/20 0.482394
rf	19/20 0.568231	19/20 **0.568231**

Table 11: SkillCraft dataset results

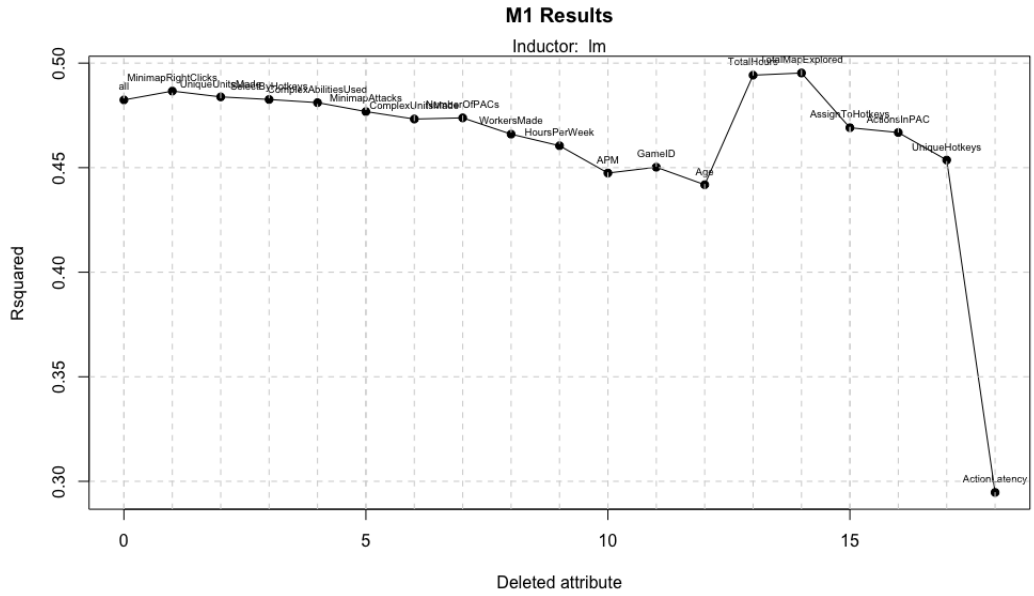

Figure 22: Best performance for SkillCraft dataset

5.2.2 Parkinson

With this dataset, the results are more adjusted to those predicted in the experimentation section; in table 12 we can see that the best performance, even the one that uses the least number of variables is achieved with Random Forest as inducers in model 1 (figure 23).

	Model 1	Model 2
lm	19/21 0.24995	19/21 0.24933
rf	2/21 **0.99783**	3/21 0.98977

Table 12: Parkinson dataset results

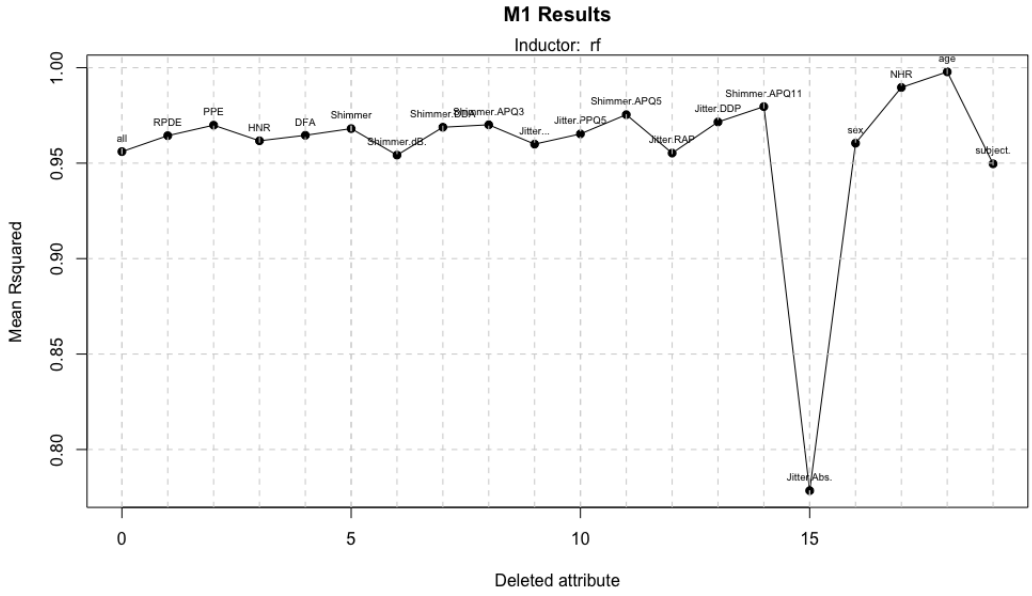

Figure 23: Best performance for Parkinson dataset

5.2.3 Breast Cancer Wisconsin

Now we will test our algorithm in a classification problem environment. In this first case, we have obtained similar results in both models using both inducers, but Linear Discriminant Analysis performs a better accuracy using less attributes than Random Forest. This performance is represented in figure 24.

	Model 1	Model 2
lda	10/11 0.9628	5/11 0.9642
rf	9/11 **0.9703**	10/11 0.9693

Table 13: Breast cancer Wisconsin dataset results

5.2.4 Ionosphere

This dataset is the largest in terms of attributes. If we look at table 14, Random Forest uses less attributes to achieve the best inducers performance in both models; if we discard

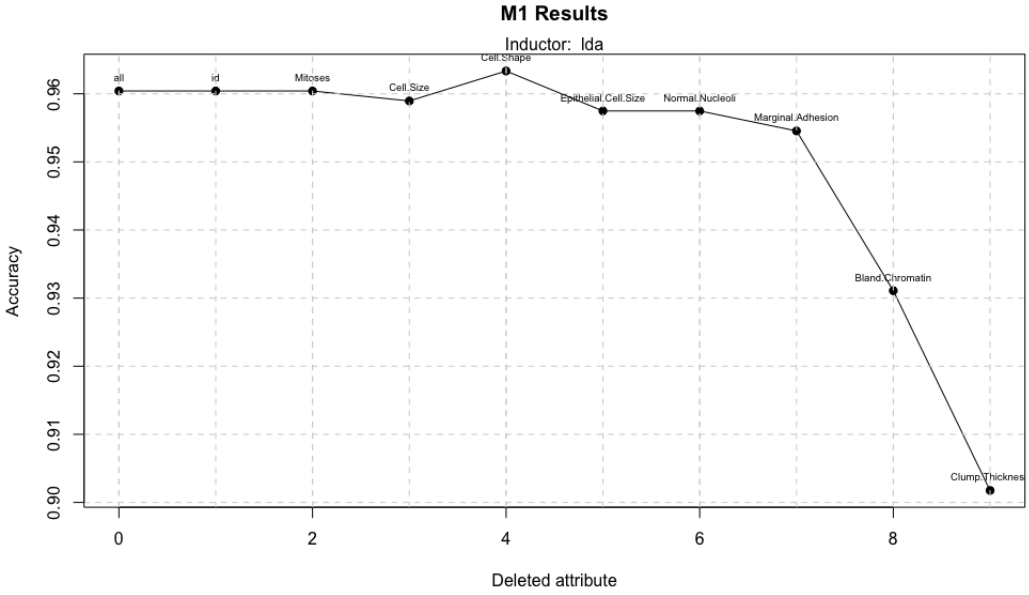

Figure 24: Best performance for Breast Cancer Wisconsin dataset

LDA as inducers for this specific example, we can conclude that Model 1 is better than Model 2 since it uses 18 of 34 attributes and it obtains a little but better score.

To compare both models we plot each best performance in figures 25 and 26.

	Model 1	Model 2
lda	25/34 0.87142	28/34 0.8687
rf	18/34 **0.9419**	20/34 0.9397

Table 14: Ionosphere dataset results

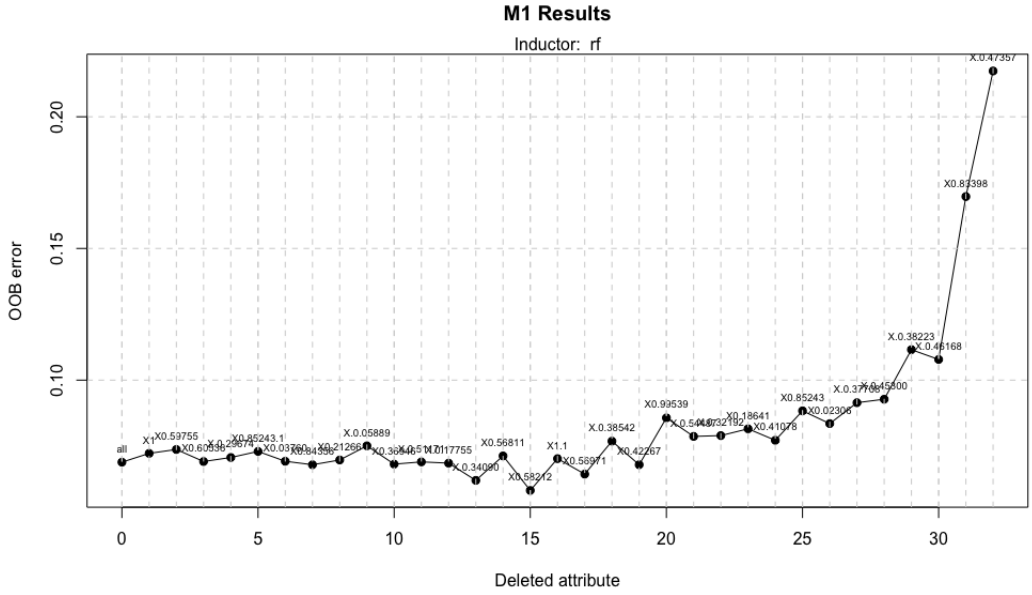

Figure 25: Best performance for Ionosphere dataset

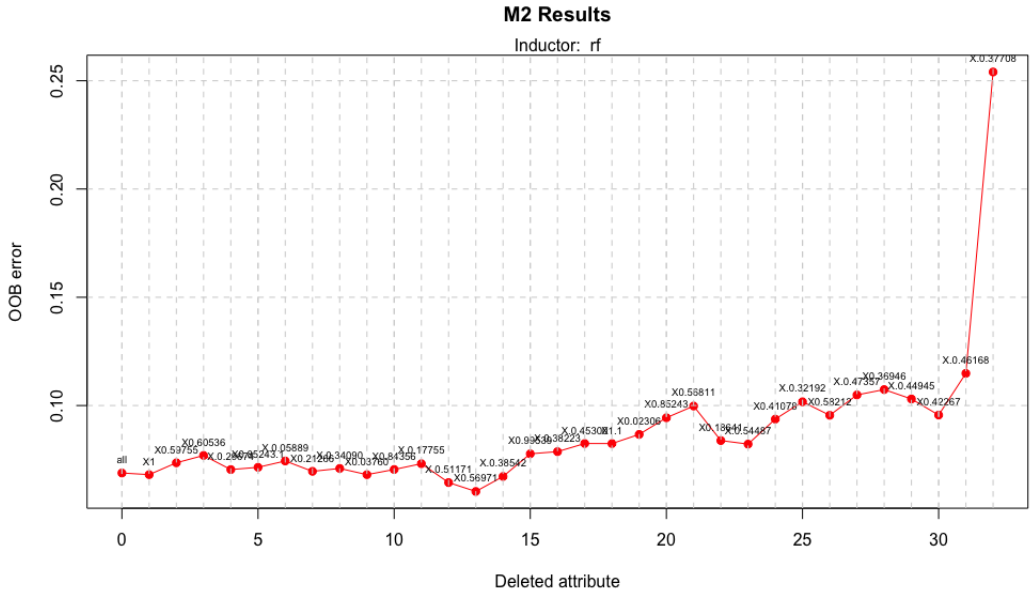

Figure 26: Best performance for Ionosphere dataset using Model 2

5.3 A comparison with filter and wrapper methods

Once we develop and evaluate our models its time to compare it with some other feature selection methods.

In next tables we present the data we have obtained performing a feature selection with a *correlation filter*, that finds attribute subset using correlation and entropy measures for continuous and discrete data, and a wrapper method using Linear Regression, both with

synthetic regression datasets we created in section 4.

These data has to be compared with the experiments we did in previous subsections. The only problem is that we don't know which attributes is removed at each iteration so we don't calculate Kendall's correlation coefficient; correlation filter gives us the best attributes and the wrapper filter checks for each subset but we can't extract a vector of removed attributes. In conclusions we will define what happens and what we expect with these data.

Dataset size	Num. Attr.	Best R^2	Execution time [s]
100	4/31	0.69668	0.83124
200	4/31	0.72956	0.88516
400	4/31	0.74339	0.71084
800	4/31	0.74229	0.73792
1600	4/31	0.75249	0.77093

Table 15: Correlation filter results

Dataset size	Num. Attr.	Best R^2	Execution time [s]
100	6/31	0.75075	1.43315
200	7/31	0.73853	1.68128
400	11/31	0.74691	1.81038
800	7/31	0.74820	3.14481
1600	11/31	0.75432	4.86907

Table 16: Wrapper filter results

6 Conclusions

This work has presented various models to perform feature selection. The slightly difference from existing approach is that it uses feature weighting algorithms to improve the performance instead of pure feature selection algorithms.

Our main goal was to develop a new feature section algorithm with iterative feature weighting methods. Our idea was that every model we develop increases the previous one in terms of performance.

In the experimental section we can see that using artificial or synthetic data can help us to work in a controlled environment in order to test and validate the work. If we take a look to model 1 experiments, we can see that we achieved the best results when the dataset size is bigger. For example, in regression, using a dataset with 400 instances we achieved an R^2 of 0.74399 and the algorithms uses 13 of 31 attributes, while in a dataset of 1600 instances we achieve an R^2 of 0.87964 with only 6 of 31 attributes.

It's important to remark that is very important to choose a good inducers. For example, in real data experiments, we can see that using Ionosphere dataset, choosing Linear Discriminant Analysis as inducers is a bad choice because we obtained an accuracy of 0.87142 using 25 of 31 attributes, while using Random Forest we obtain an accuracy of 0.9419 with 18 of 31 attributes. There are other cases, for example the experiment with SkillCraft dataset. Here we obtain a better R^2 with Random Forest, but the model uses 19 of 20 attributes. Using Linear Regression we obtained a slightly smaller R^2 but it only uses 5 of 30 attributes.

We also use Kendall's correlation coefficient to evaluate how well a model classifies the attributes. In regression, the best result is obtained using model 2 and Random Forest as inducers. In classification, the best result is obtained using model 1 and Linear Discriminant analysis. The main conclusions we can extract after this work, taking into account the Kendall's correlation coefficient, is that model 1 works better in a classification environment where we use the original Relief algorithm and model 2 work better in a regression environment.

If we take a look to the last experiment with a correlation filter and a wrapper filter, we could see that our models are only better in one case, when the dataset is the biggest and we use Random Forest as inducer. Despite this, we can conclude that our models are not good as we expect because we gain in terms of R^2 to other ones by a very little difference and the time execution is much bigger.

I would like to comment that the hypothesis we propose in the beginning, where we said model 2 is an improvement of model 1, is not achieved as we expected because we can't see a bigger qualitative leap. There are a lot of parameters that can be chosen by the user, depends on the parameters that are most important to him. Sometimes can be more important the number of attributes, and that means obtain a worst performance result; or sometimes can be more important the time execution.

6.1 Future work

Future work could include try another algorithm for Feature Weighting such as Simba that uses the attribute importance to calculate the distance between instances in order to perform the classification. To increase the experimental section, it can be useful to vary the sample size in datasets or increase the number of attributes. As we could see, Random Forest works very well as inducer, so another option could be tune the algorithm to increase the performance.

In synthetic datasets we worked with relevant, redundant and irrelevant features. It would be interesting to introduce corrupt data with missing values to see how the algorithm deals with it.

References

[Aha und Bankert 1996] Aha, David W. ; Bankert, Richard L.: A comparative evaluation of sequential feature selection algorithms. (1996), S. 199–206

[Doak 1992] Doak, Justin: An Evaluation of Feature Selection Methodsand Their Application to Computer Security. (1992)

[James u. a. 2014] James, Gareth ; Witten, Daniela ; Hastie, Trevor ; Tibshirani, Robert: *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. – ISBN 1461471370, 9781461471370

[Kira und Rendell 1992b] Kira, Kenji ; Rendell, Larry A.: A practical approach to feature selection. In: Sleeman, D. (Hrsg.) ; Edwards, P. (Hrsg.): *Machine Larning: Proceedings of International Conference (ICML'92)*. Morgan Kaufmann, 1992b, S. 249–256

[Kononenko 1994] Kononenko, Igor: Estimating attributes: analysis and extensions of Relief. In: Raedt, L. D. (Hrsg.) ; Bergadano, F. (Hrsg.): *Machine Learning: ECML-94*. Springer Verlag, 1994, S. 171–182

[Kuhn 2017] Kuhn, Max: *Classification and Regression Training*. : , 2017. – URL `https://github.com/topepo/caret/`. – Version 6.0-78

[Leisch und Dimitriadou 2010] Leisch, Friedrich ; Dimitriadou, Evgenia: *mlbench: Machine Learning Benchmark Problems*. : , 2010. – R package version 2.1-1

[Liaw und Wiener. 2015] Liaw, Andy ; Wiener., Matthew: *Breiman and Cutler's Random Forests for Classification and Regression*. : , 2015. – URL `https://www.stat.berkeley.edu/~breiman/RandomForests/`. – Version 4.6-12

[Mark B. und C. 2013] Mark B., Andrew H. ; C., Bill: *SkillCraft1 Master Table Dataset Data Set*. 2013. – URL `http://archive.ics.uci.edu/ml/datasets/skillcraft1+master+table+dataset#`

[Marko Robnik-Sikonja 2017] Marko Robnik-Sikonja, Petr S.: *Classification, Regression and Feature Evaluation*. : , 2017. – URL `https://CRAN.R-project.org/package=CORElearn`. – Version 1.51.2

[Ripley 2017] Ripley, Brian: *Support Functions and Datasets for Venables and Ripley's MASS*. : , 2017. – URL `http://www.stats.ox.ac.uk/pub/MASS4/`. – Version 7.3-48

[Robnik-Sikonja und Kononenko 1997] Robnik-Sikonja, Marko ; Kononenko, Igor: An adaptation of Relief for attribute estimation in regression. In: Fisher, D. H. (Hrsg.): *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*. Morgan Kaufmann, 1997, S. 296–304

[Robnik-Sikonja und Kononenko 2003] Robnik-Sikonja, Marko ; Kononenko, Igor: Theoretical and Empirical Analysis of ReliefF and RReliefF. In: *Machine Learning* 53 (2003), S. 23–69

[Sigillito 1989] Sigillito, Vince: *Ionosphere*. 1989. – URL `https://archive.ics.uci.edu/ml/datasets/ionosphere`

[T. und L. 2009] T., Athanasios ; L., Max: *Oxford Parkinson's Disease Telemonitoring Dataset*. 2009. – URL `https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring`

[Team 2008] Team, R Development C.: *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing (Veranst.), 2008. – URL `http://www.R-project.org`. – ISBN 3-900051-07-0

[Wolberg 1995] Wolberg, Dr. William H.: *Breast Cancer Wisconsin (Diagnostic)*. 1995. – URL `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`

Appendix

Appendix A Model 2

```
model2 <- function(data, inducers, data_type){
  start_time <- Sys.time()

  set.seed(24)

  n <- ncol(data)
  nplot <- n

  # Regression Problem
  if (class(data[, n]) == "numeric" || class(data[, n]) == "
      integer") {
    print("Regression")

    # Relief
    library(CORElearn)
    formula <- as.formula(paste(names(data)[n], "~ ."))
    result <- attrEval(n, data, estimator="RReliefFequalK",
        ReliefIterations=1000)
    res <- as.data.frame(t(result))

    sort.data <- data[c(names(sort(res, decreasing = TRUE)),
        names(data)[nplot])]
    relief.names <- colnames(sort(res, decreasing = FALSE))

    inducers.result <- rep(NA, n-1)
    data.names <- rep(NA, n-1)

    # Linear regression
    if (inducers == "lm") {
      print("Linear Regression")

      # Cross validation
      library(caret)
      train_control <- trainControl(method="cv", number=10)

      # Complete dataset
      model <- train(formula, data=sort.data, trControl=train_
          control, method="lm")
      inducers.result[1] <- as.numeric(model$results[3])
      data.names[1] <- "all"

      # Delete one attribute at a time
      for (i in 1:(n-2)) {
        sort.data <- sort.data[, -(n-1)]
```

```
      data.names[i+1] <- relief.names[1]
      n <- ncol(sort.data)

      # Calculate Relief classification
      result <- attrEval(n, sort.data, estimator="
          RReliefFequalK", ReliefIterations=1000)
      res <- as.data.frame(t(result))

      sort.data <- data[c(names(sort(res, decreasing = TRUE))
          , names(data)[nplot])]
      relief.names <- colnames(sort(res, decreasing = FALSE))

      # Cross validation
      model <- train(formula, data=sort.data, trControl=train
          _control, method="lm")
      inducers.result[i+1] <- as.numeric(model$results[3])

      print(i)
    }

    ylabel <- "R-squared"
  }

  # Random forest
  else {
    print("Random Forest")

    library("randomForest")
    # Complete dataset
    rf <- randomForest(formula, data=sort.data, ntree=150,
        proximity=FALSE)
    mR-squared <- mean(rf$rsq)
    inducers.result[1] <- as.numeric(mR-squared)
    data.names[1] <- "all"

    # Delete one attribute at a time
    for (i in 1:(n-2)) {
      sort.data <- sort.data[, -(n-1)]
      data.names[i+1] <- relief.names[1]
      n <- ncol(sort.data)

      # Calculate Relief classification
      result <- attrEval(n, sort.data, estimator="
          RReliefFequalK", ReliefIterations=1000)
      res <- as.data.frame(t(result))

      sort.data <- data[c(names(sort(res, decreasing = TRUE))
          , names(data)[nplot])]
      relief.names <- colnames(sort(res, decreasing = FALSE))
```

```
    # Random Forest
    rf <- randomForest(formula, data=sort.data, ntree=150,
        proximity=FALSE)
    mR-squared <- mean(rf$rsq)
    inducers.result[i+1] <- as.numeric(mR-squared)

    print(i)
  }

  ylabel <- "Mean R-squared"
}

if (data_type == "artificial") {
  # Kendall correlation
  print("Kendall Correlation")
  truth <- c(1, 1, 3, 4, 5, 30, 30, 30, 30, 30, 30, 30, 30,
      30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
      30, 10, 10, 5)
  names(truth) <- names(data[-31])
  solution <- solution <- c(1, 1, 3, 4, 5, 5, 10, 10, 30,
      30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
      30, 30, 30, 30, 30, 30, 30, 30)
  names(solution) <- rev(c(data.names[2:length(data.names)
      ], names(sort.data[1])))
  solution <- solution[names(truth)]

  m <- cbind(truth, solution)
  print(cor(m, method = "kendall"))
}

# Add better attribute to names.full
names.full <- c(data.names[2:length(data.names)], relief.
    names)
# Print which attributes we need to perform the best
    inducers result
attributes <- names.full[which.max(inducers.result):length(
    names.full)]
print(paste("Best performance: ", max(inducers.result)))
print(paste("Number of attributes needed for best
    performance:", length(attributes), "of", nplot))

# Plot results
plot(0:(nplot-2), inducers.result, type="o", main="M2
    Results", ylab = ylabel, xlab = "Deleted attribute",
    col="red",
      pch=21, bg="red")
text(0:(nplot-2), inducers.result, data.names, cex=0.6, pos
    =3)
```

```
  mtext(paste("inducers: ", inducers), side=3)
  grid(lty=2)
  abline(v=0:(nplot-2), lty=2, col='lightgrey')
}

# Classification Problem
else {
  print("Classification")

  # Relief
  library(CORElearn)
  formula <- as.formula(paste(names(data)[n], "~ ."))
  result <- attrEval(formula, data, estimator="Relief",
      ReliefIterations=1000)
  res <- as.data.frame(t(result))

  sort.data <- data[c(names(sort(res, decreasing = TRUE)),
      names(data)[nplot])]
  relief.names <- colnames(sort(res, decreasing = FALSE))

  inducers.result <- rep(NA, n-1)
  data.names <- rep(NA, n-1)

  if (inducers == "lda") {
    print("Linear Discriminant Analysis")

    library(MASS)
    data.lda <- lda(formula, data=sort.data, CV=TRUE)

    tab <- table(data[, n], data.lda$class)
    acc <- sum(diag(prop.table(tab)))

    inducers.result[1] <- as.numeric(acc)
    data.names[1] <- "all"

    # Delete one attribute at a time
    for (i in 1:(n-2)) {
      sort.data <- sort.data[, -(n-i)]
      data.names[i+1] <- relief.names[1]

      # Calculate Relief classification
      result <- attrEval(formula, sort.data, estimator="
          Relief", ReliefIterations=1000)
      res <- as.data.frame(t(result))

      sort.data <- data[c(names(sort(res, decreasing = TRUE))
          , names(data)[nplot])]
      relief.names <- colnames(sort(res, decreasing = FALSE))
```

```
    # LDA
    data.lda <- lda(formula, data=sort.data, CV=TRUE)

    tab <- table(data[, n], data.lda$class)
    acc <- sum(diag(prop.table(tab)))
    inducers.result[i+1] <- as.numeric(acc)
    print(i)
  }

  ylabel <- "Accuracy"
}

else {
  print("Random Forest")

  library("randomForest")
  # Complete dataset
  rf <- randomForest(formula, data=sort.data, ntree=150,
      proximity=FALSE)
  #acc <- sum(diag(prop.table(rf$confusion)))
  oob.err <- sum(rf$err.rate[, 1])/150
  inducers.result[1] <- as.numeric(oob.err)
  data.names[1] <- "all"

  # Delete one attribute at a time
  for (i in 1:(n-2)) {
    sort.data <- sort.data[, -(n-i)]
    data.names[i+1] <- relief.names[1]

    # Calculate Relief classification
    result <- attrEval(formula, sort.data, estimator="
        Relief", ReliefIterations=1000)
    res <- as.data.frame(t(result))

    sort.data <- data[c(names(sort(res, decreasing = TRUE))
        , names(data)[nplot])]
    relief.names <- colnames(sort(res, decreasing = FALSE))

    # Random Forest
    rf <- randomForest(formula, data = sort.data, ntree
        =150, proximity=FALSE)
    #acc <- sum(diag(prop.table(rf$confusion)))
    oob.err <- sum(rf$err.rate[, 1])/150
    inducers.result[i+1] <- as.numeric(oob.err)
    print(i)
  }

  ylabel <- "OOB error"
}
```

```
  if (data_type == "artificial") {
    # Kendall correlation
    print("Kendall Correlation")
    truth <- c(1, 1, 3, 4, 5, 10, 10, 10, 10, 10)
    names(truth) <- names(data[, -11])
    solution <- c(1, 1, 3, 4, 5, 10, 10, 10, 10, 10)
    names(solution) <- rev(c(data.names[2:length(data.names)
        ], names(sort.data[1])))
    solution <- solution[names(truth)]

    m <- cbind(truth, solution)
    print(cor(m, method = "kendall"))
  }

  # Add better attribute to names.full
  names.full <- c(data.names[2:length(data.names)], relief.
      names)
  # Print which attributes we need to perform the best
      inducers result
  if (inducers == "rf") {
    attributes <- data.names[which.min(inducers.result):
        length(data.names)]
    print(paste("Best performance: ", min(inducers.result)))
  }
  else {
    attributes <- data.names[which.max(inducers.result):
        length(data.names)]
    print(paste("Best performance: ", max(inducers.result)))
  }

  print(paste("Number of attributes needed for best
      performance:", length(attributes), "of", nplot))

  # Plot results
  print("Plot Results")
  plot(0:(nplot-2), inducers.result, type="o", main="M2
      Results", ylab = ylabel, xlab = "Deleted attribute",
      col="red",
        pch=21, bg="red")
  text(0:(nplot-2), inducers.result, data.names, cex=0.6, pos
      =3)
  mtext(paste("inducers: ", inducers), side=3)
  grid(lty=2)
  abline(v=0:(nplot-2), lty=2, col='lightgrey')
}

end_time <- Sys.time()
print(end_time - start_time)
```

```
    print("End")
}
```

Appendix B Model 1 regression results

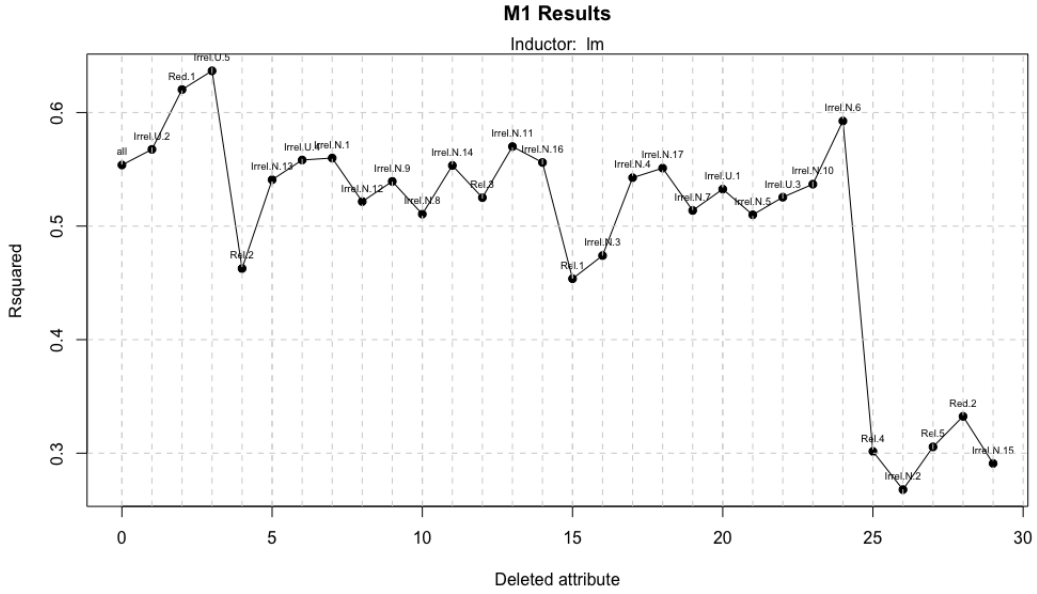

Figure 27: Model 1 performance with 100 instances dataset

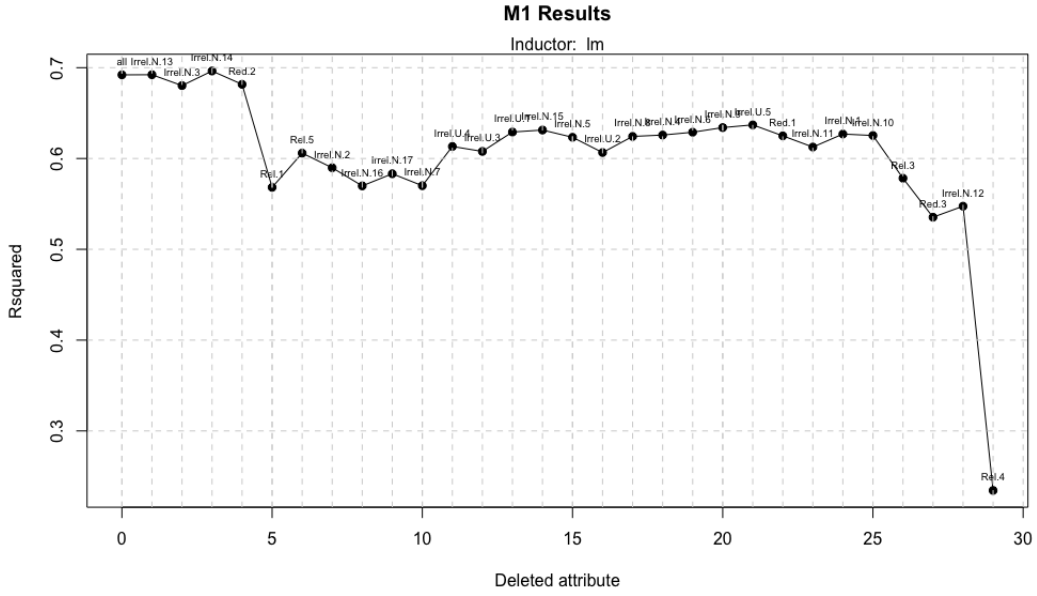

Figure 28: Model 1 performance with 200 instances dataset

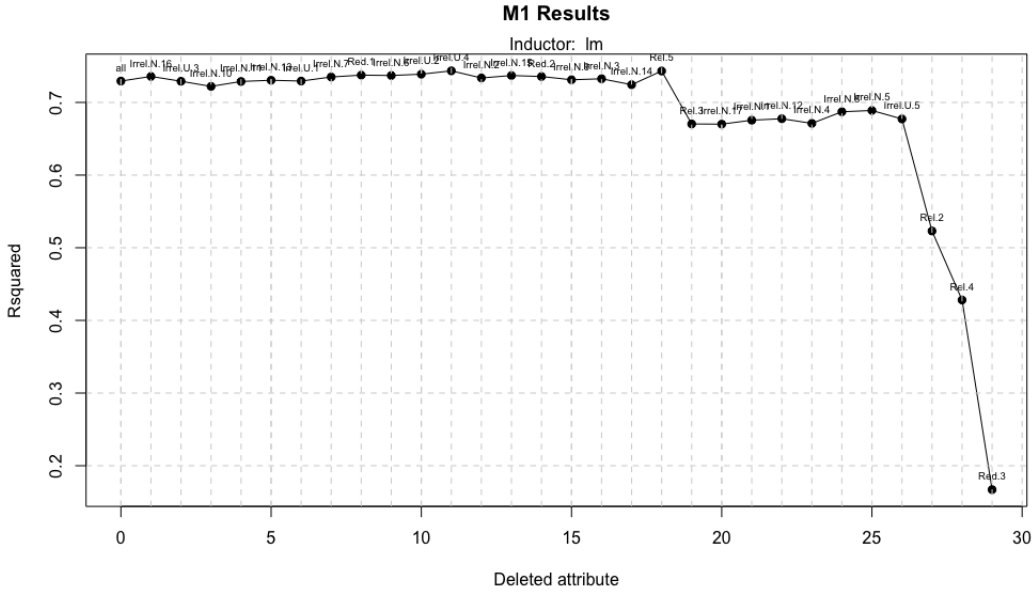

Figure 29: Model 1 performance with 400 instances dataset

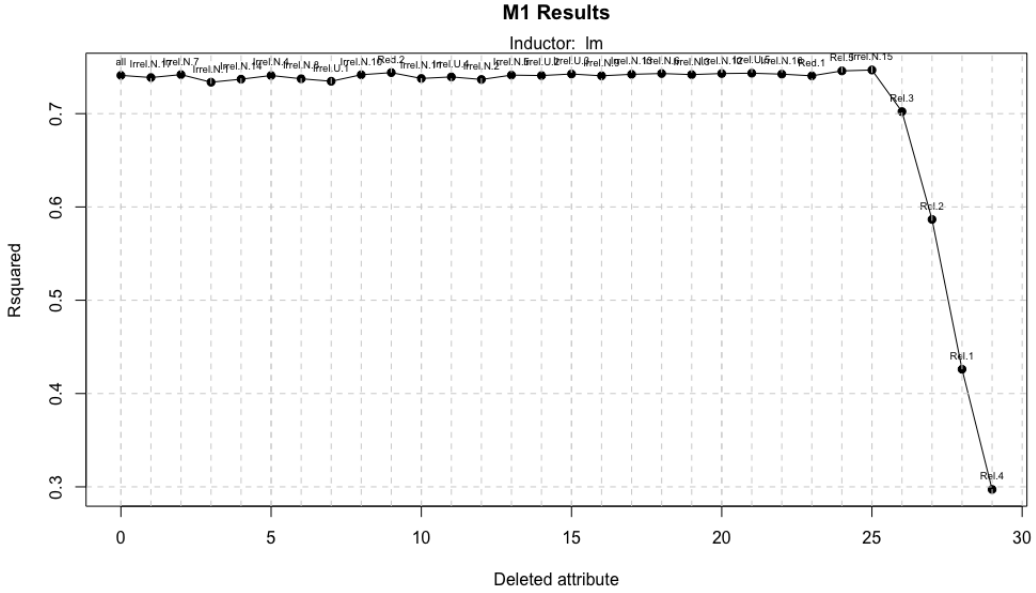

Figure 30: Model 1 performance with 800 instances dataset

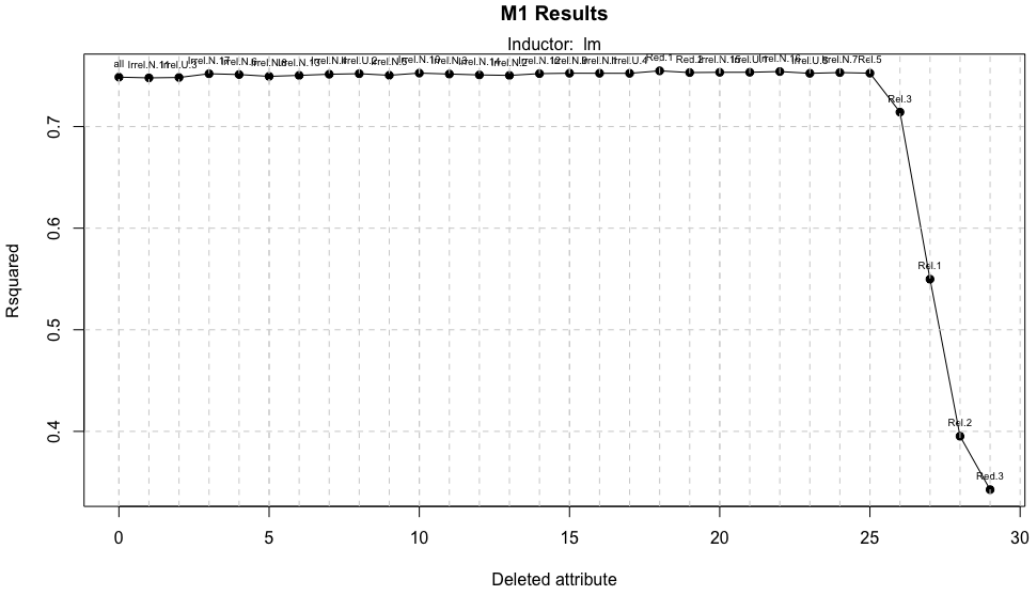

Figure 31: Model 1 performance with 1600 instances dataset

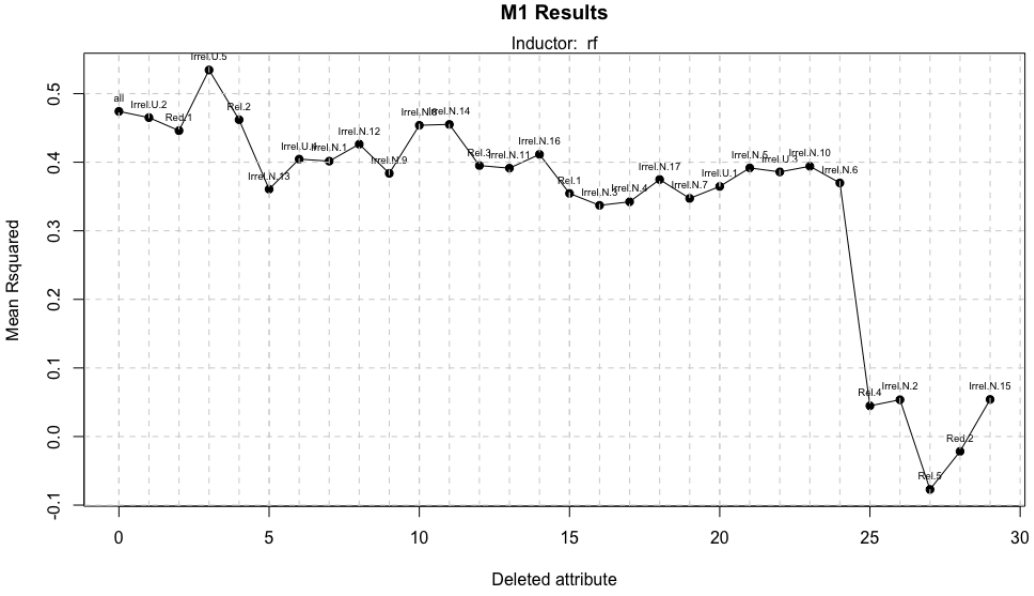

Figure 32: Model 1 performance with 100 instances dataset

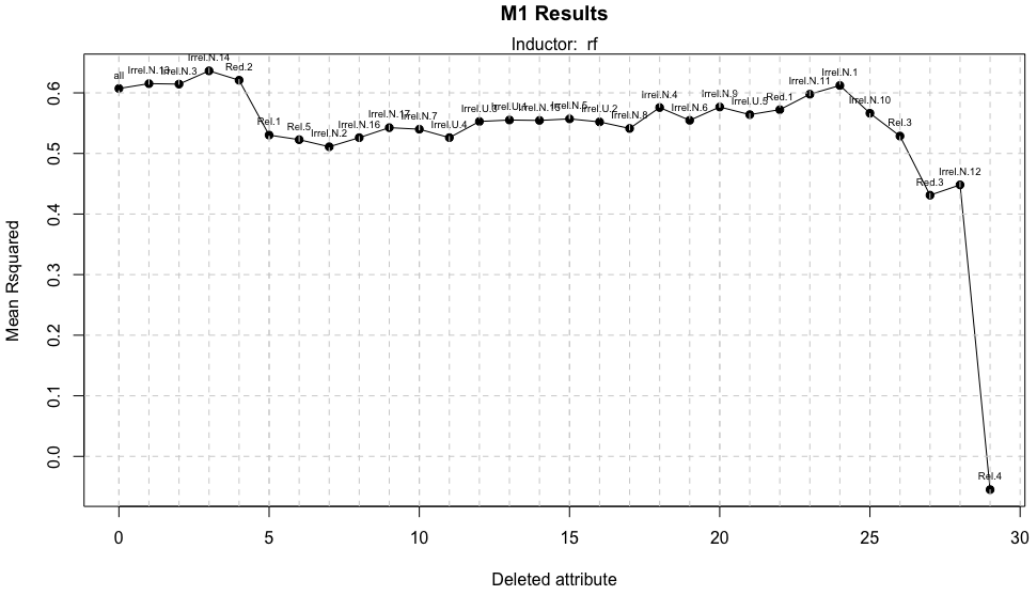

Figure 33: Model 1 performance with 200 instances dataset

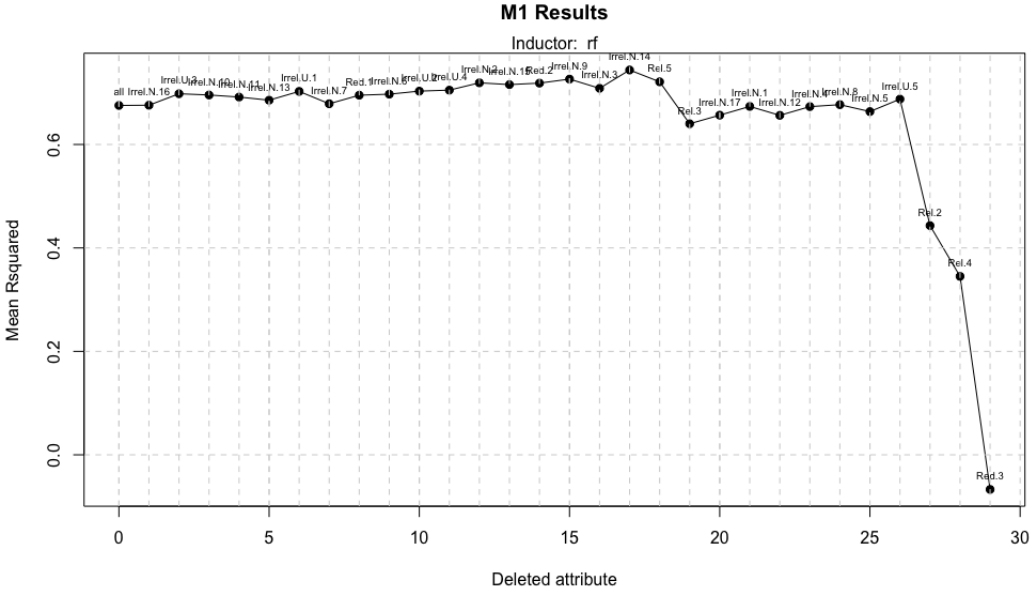

Figure 34: Model 1 performance with 400 instances dataset

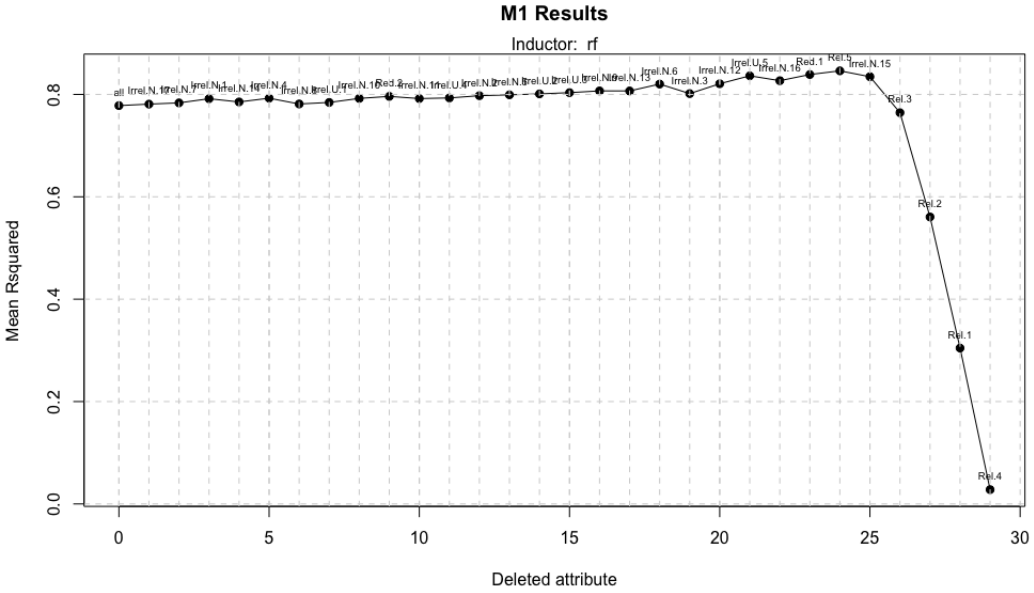

Figure 35: Model 1 performance with 800 instances dataset

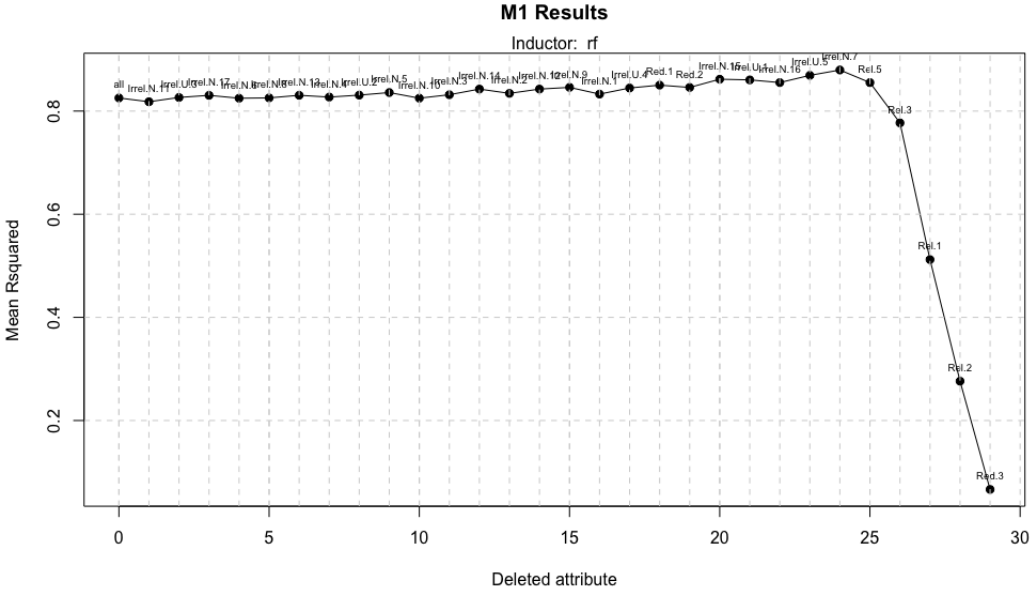

Figure 36: Model 1 performance with 1600 instances dataset

Appendix C Model 1 classification results

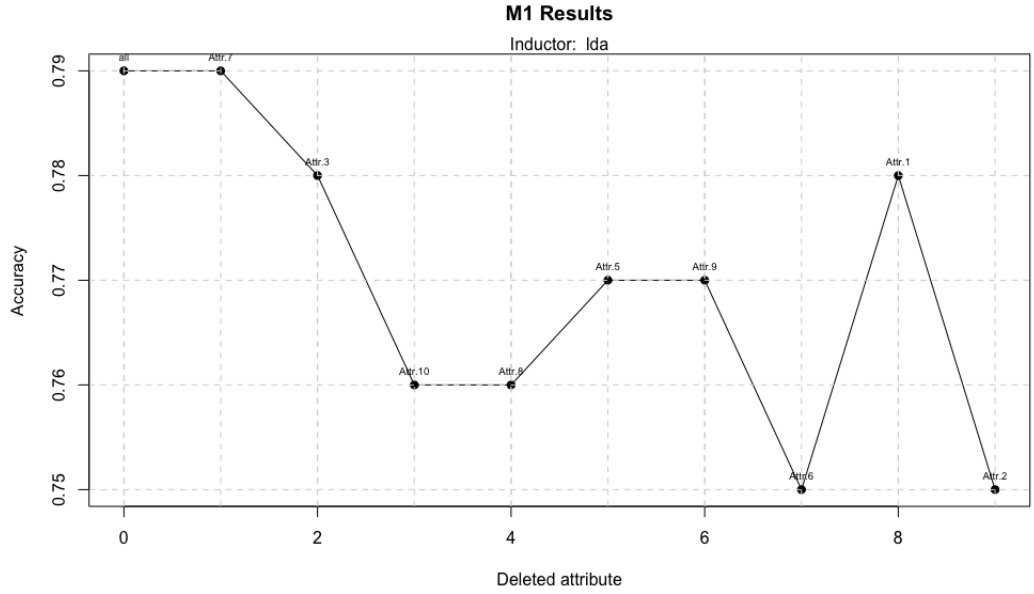

Figure 37: Model 1 performance with 100 instances dataset

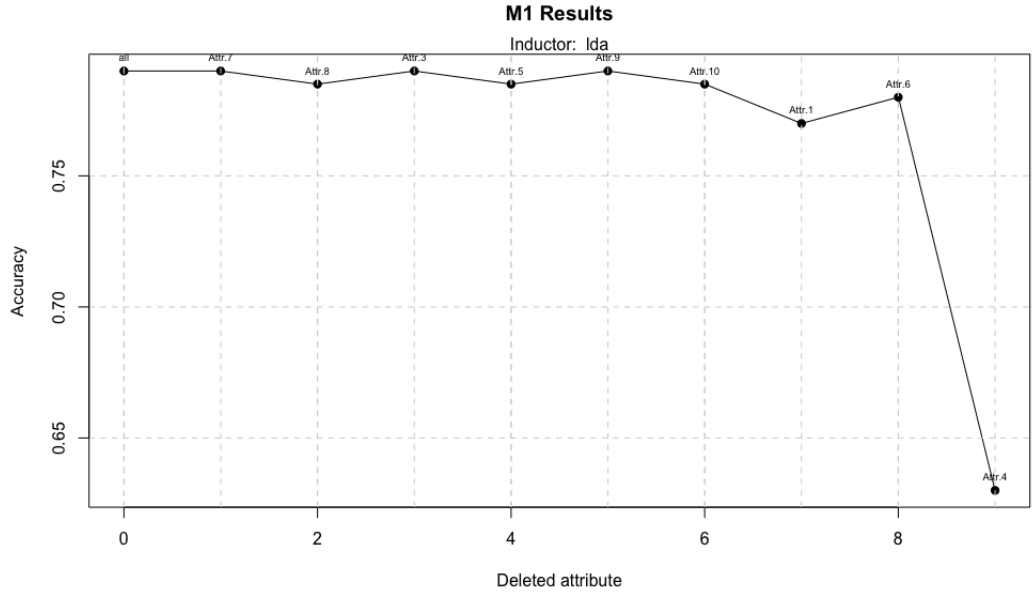

Figure 38: Model 1 performance with 200 instances dataset

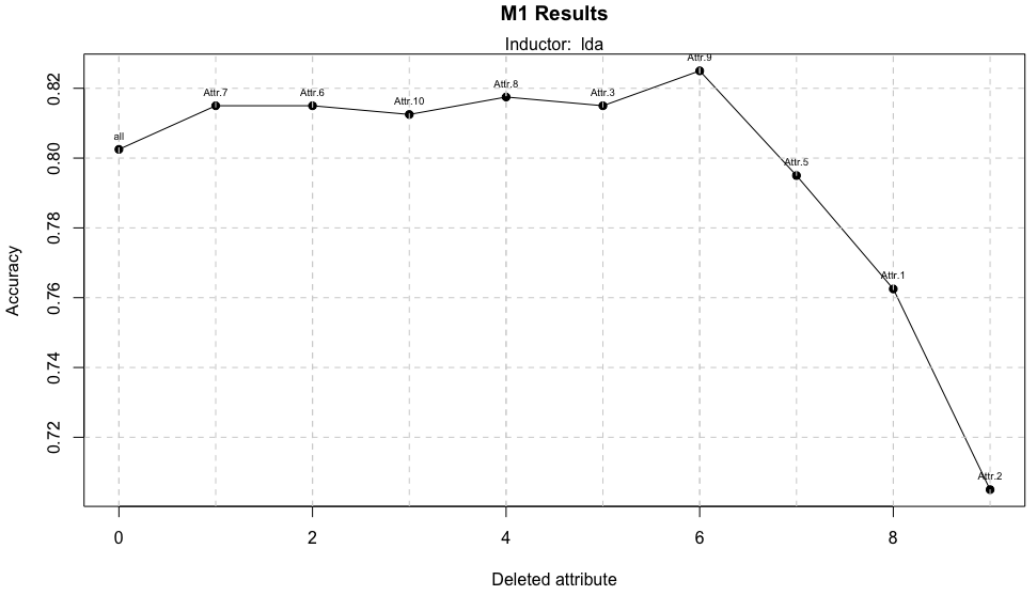

Figure 39: Model 1 performance with 400 instances dataset

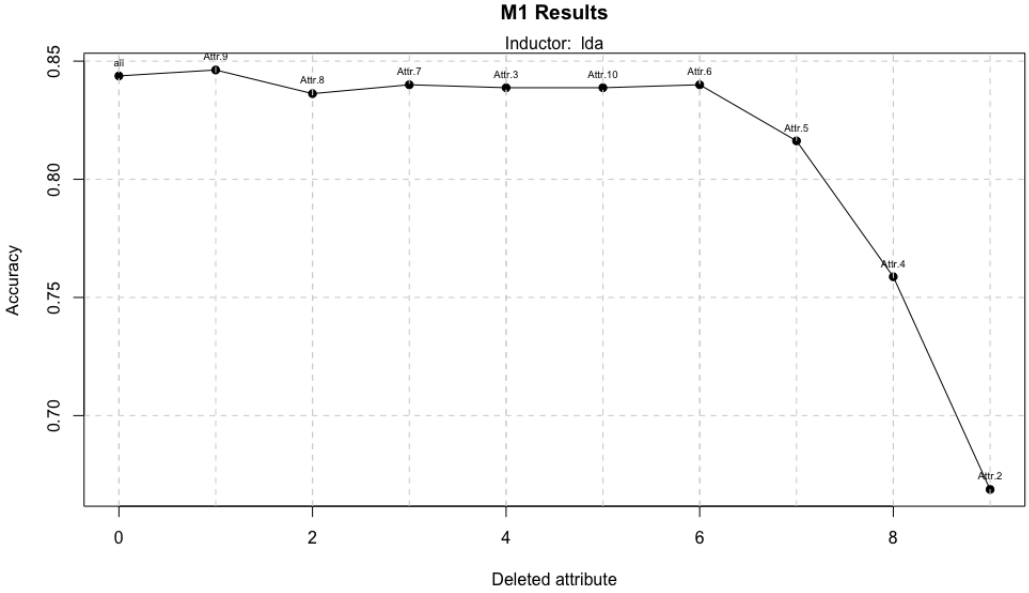

Figure 40: Model 1 performance with 800 instances dataset

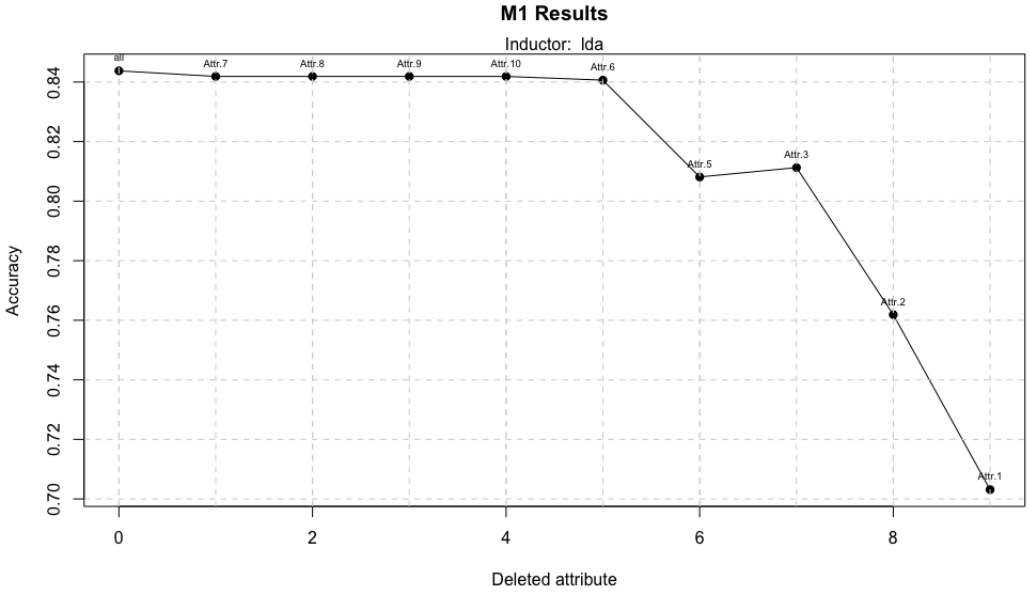

Figure 41: Model 1 performance with 1600 instances dataset

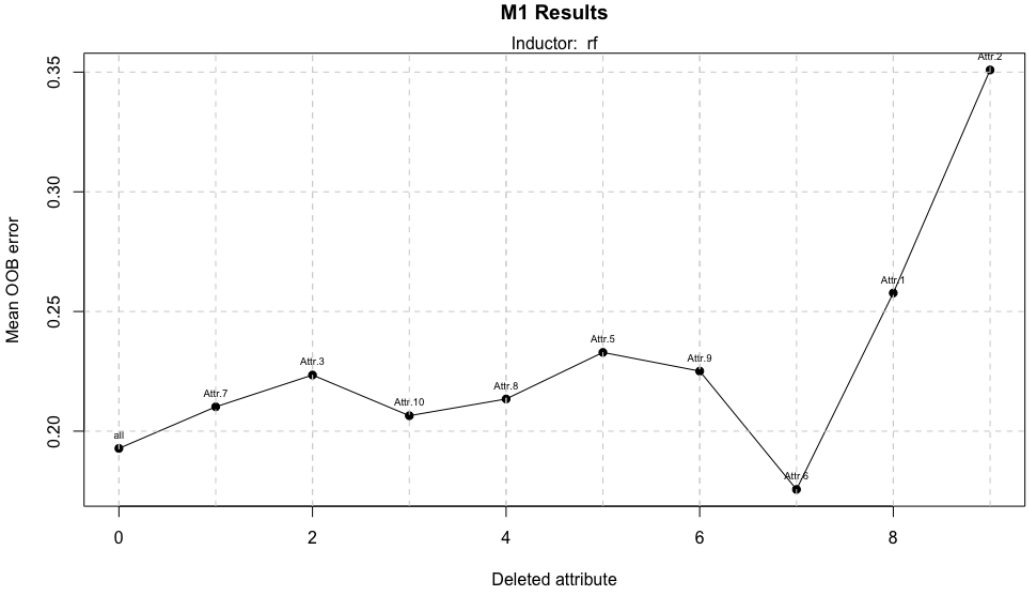

Figure 42: Model 1 performance with 100 instances dataset

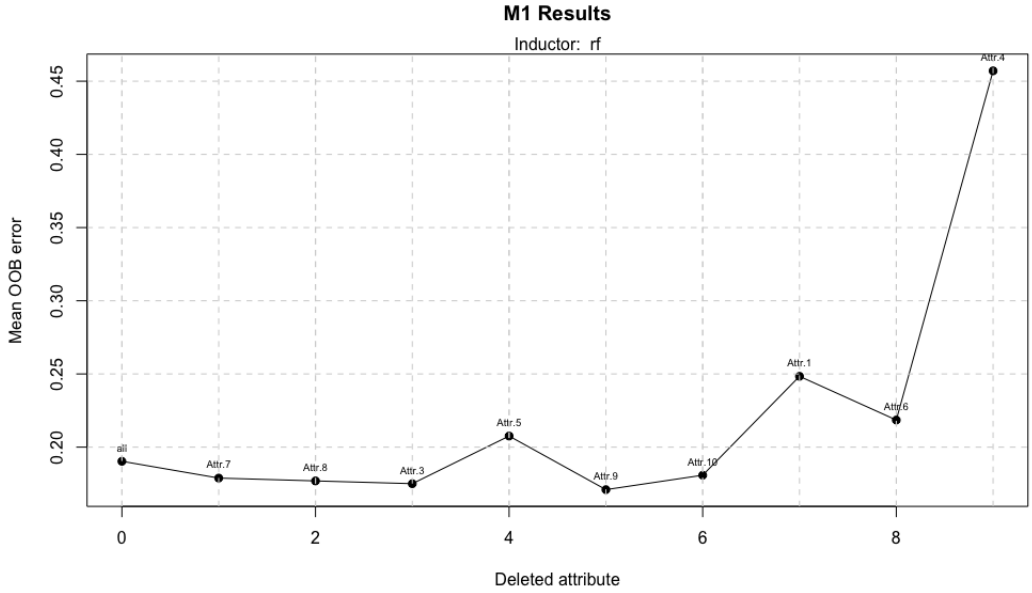

Figure 43: Model 1 performance with 200 instances dataset

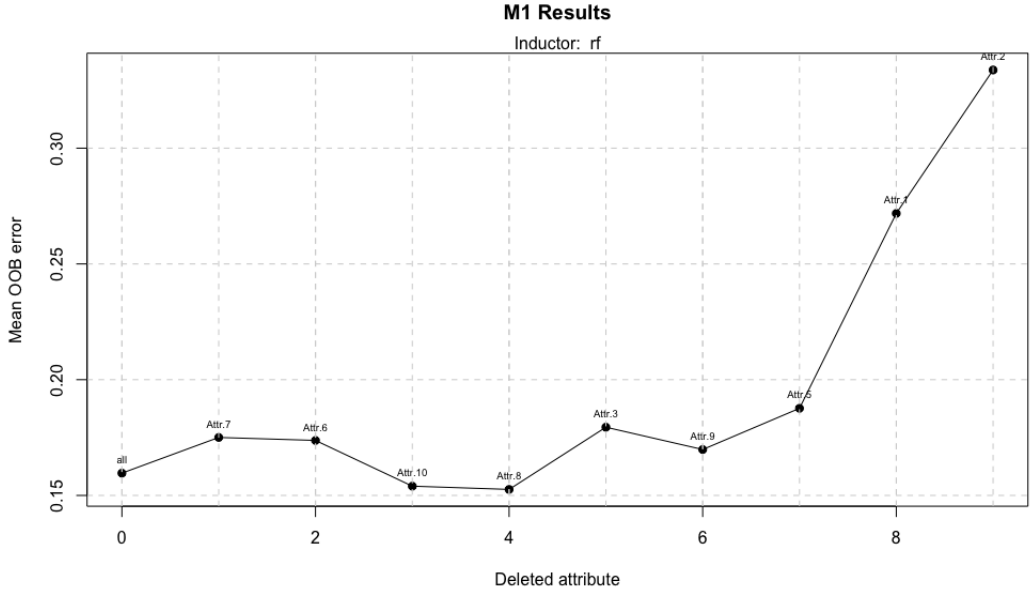

Figure 44: Model 1 performance with 400 instances dataset

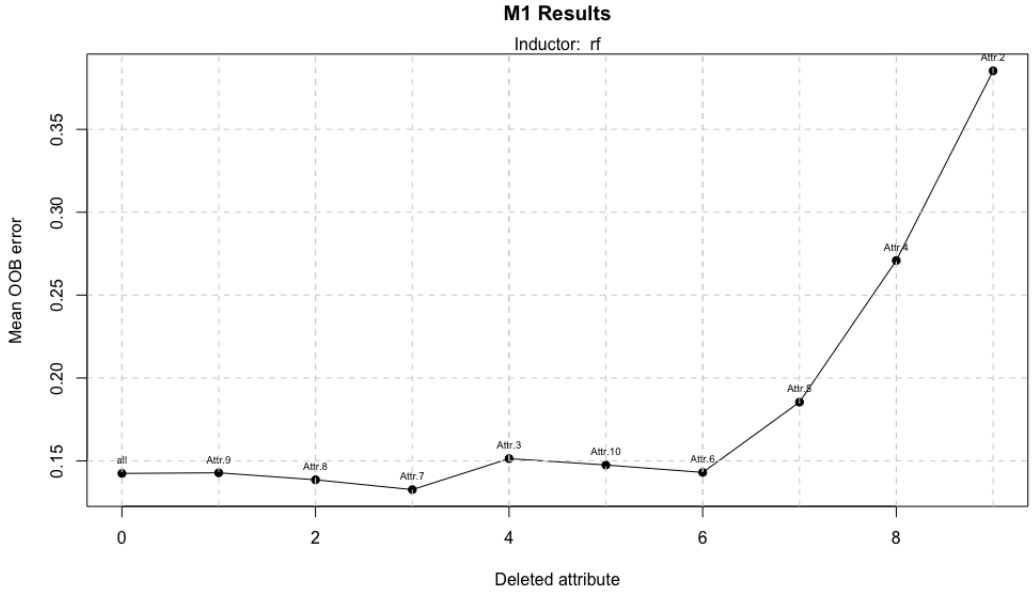

Figure 45: Model 1 performance with 800 instances dataset

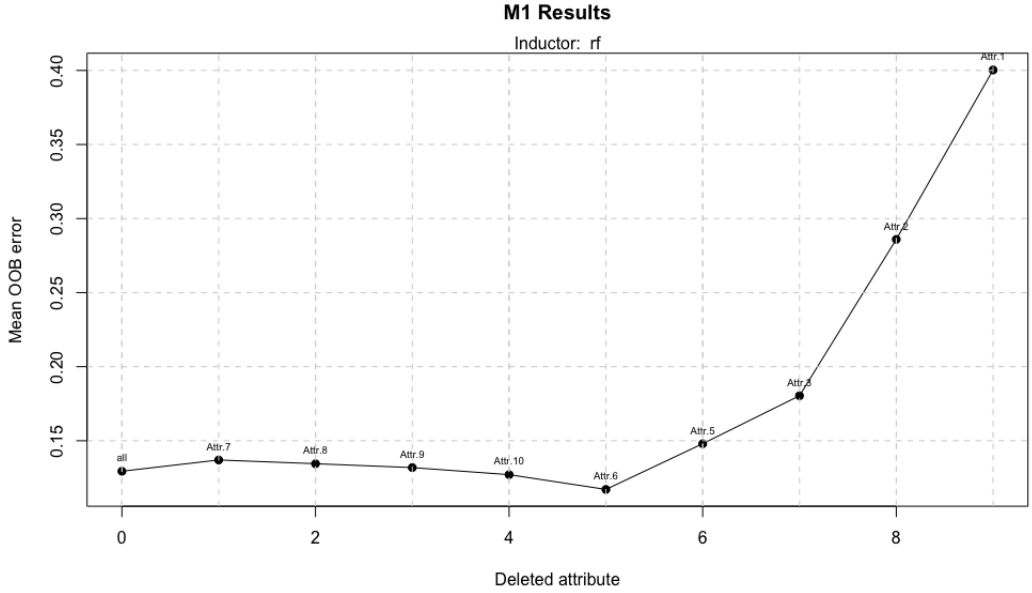

Figure 46: Model 1 performance with 1600 instances dataset

Appendix D Model 2 regression results

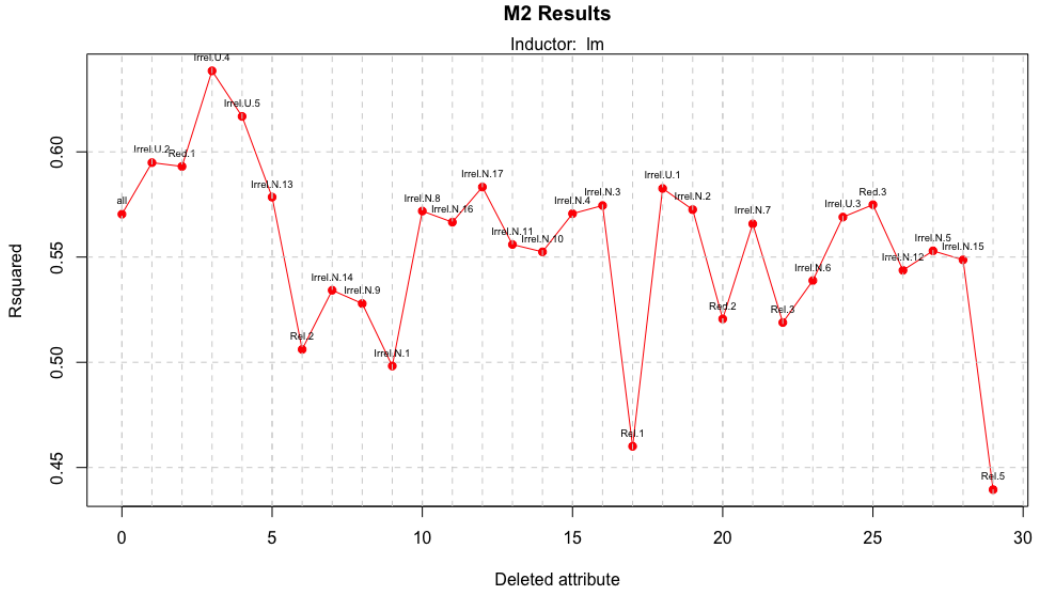

Figure 47: Model 2 performance with 100 instances dataset

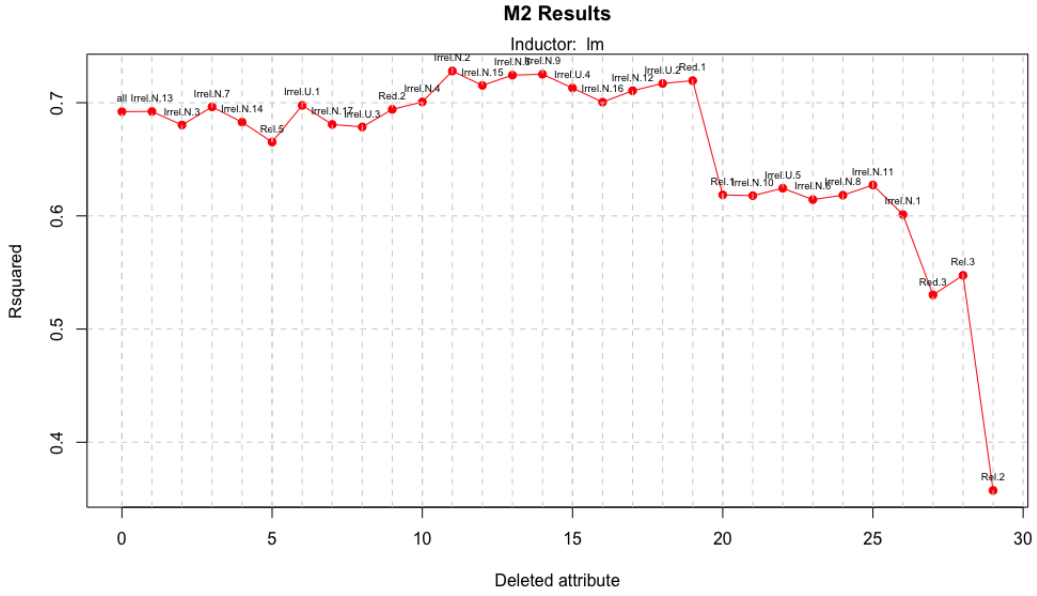

Figure 48: Model 2 performance with 200 instances dataset

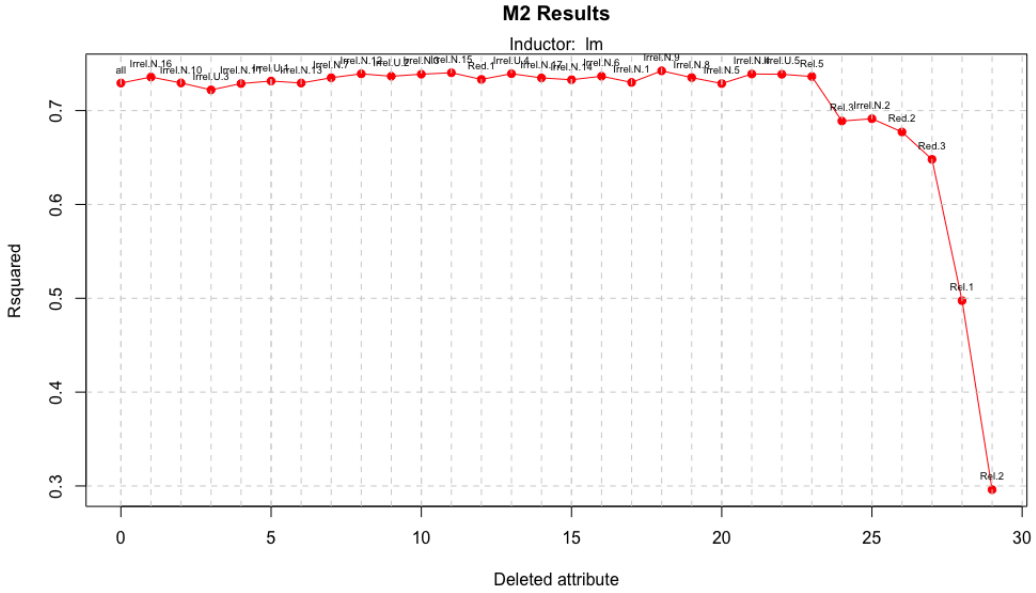

Figure 49: Model 2 performance with 400 instances dataset

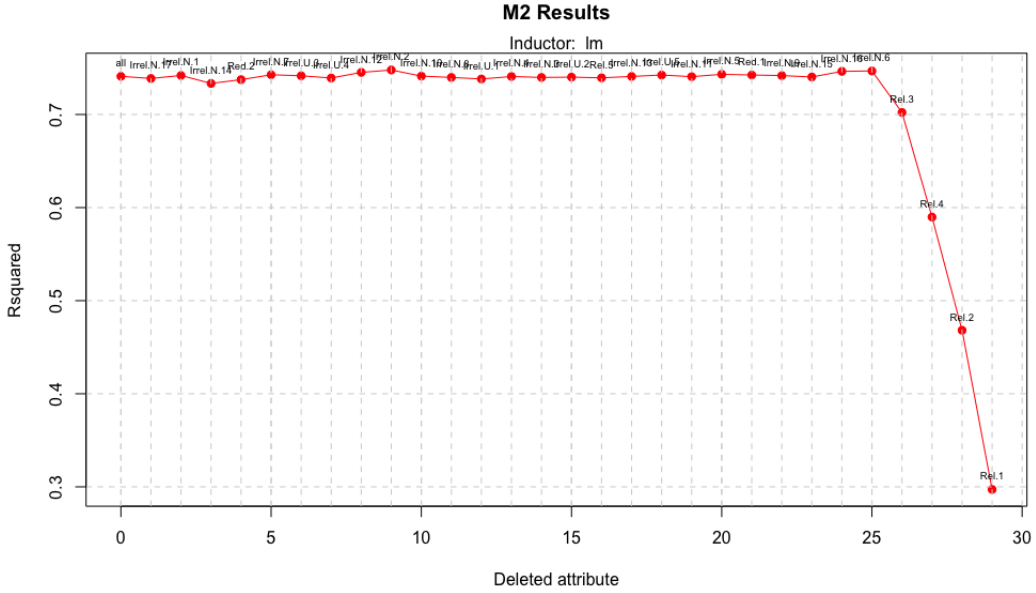

Figure 50: Model 2 performance with 800 instances dataset

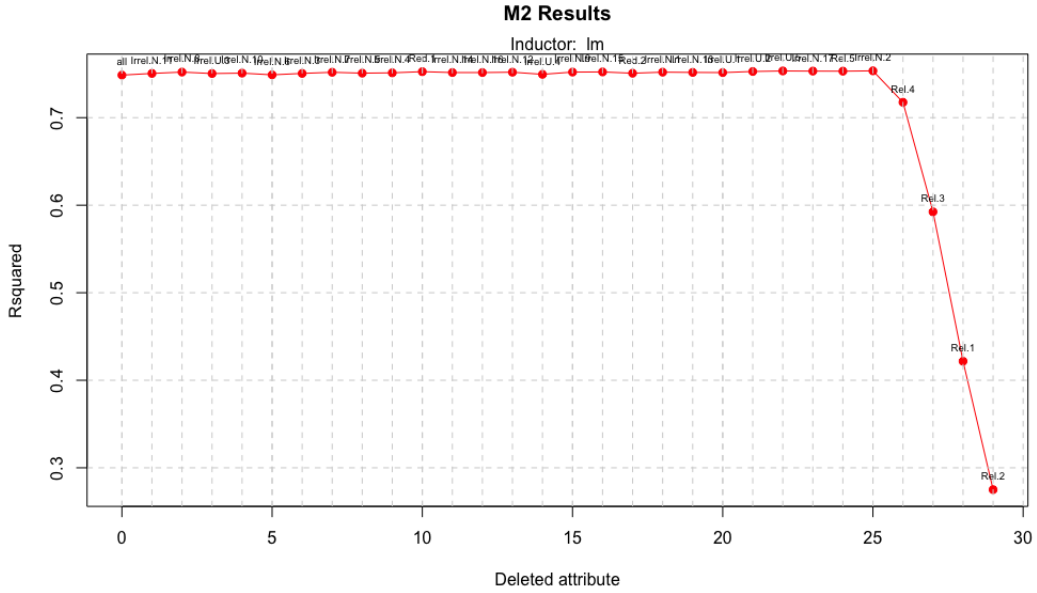

Figure 51: Model 2 performance with 1600 instances dataset

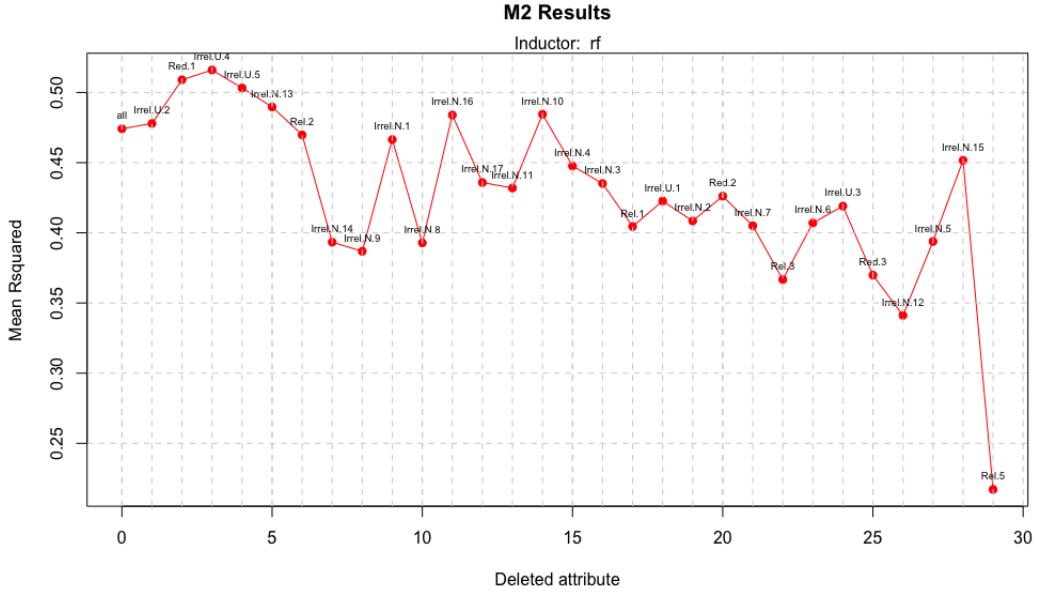

Figure 52: Model 2 performance with 100 instances dataset

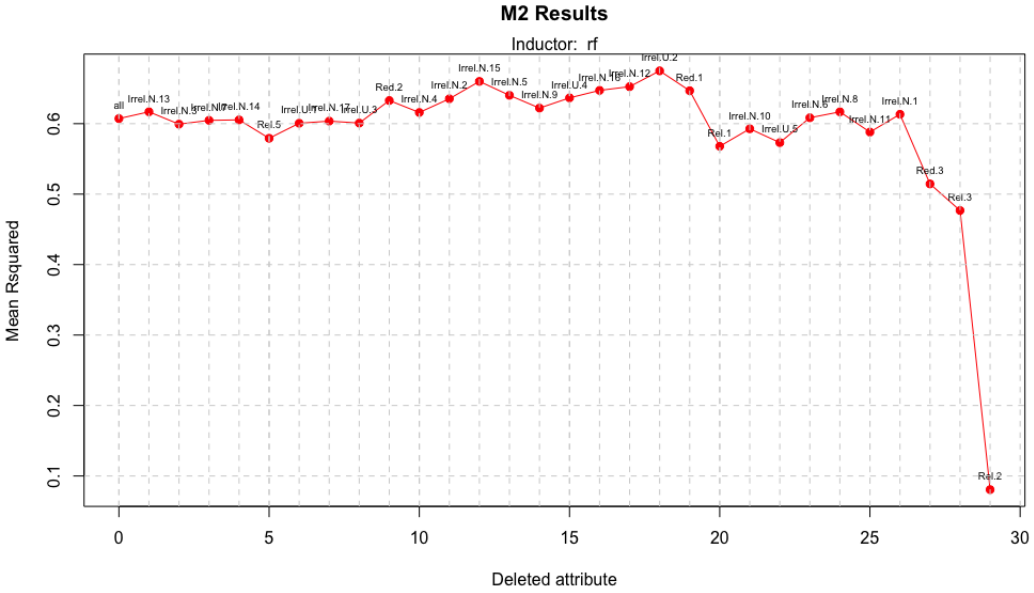

Figure 53: Model 2 performance with 200 instances dataset

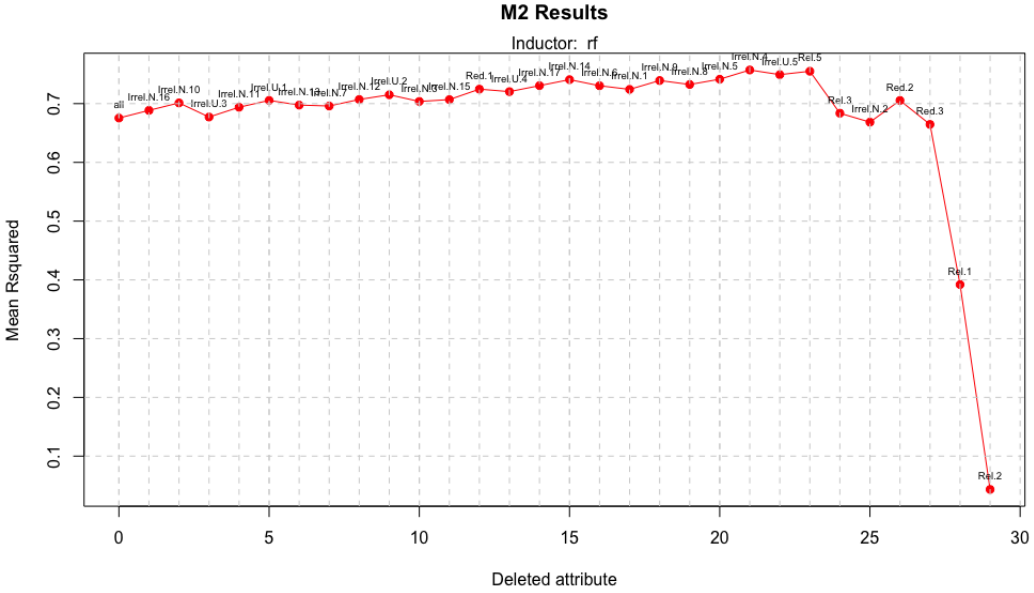

Figure 54: Model 2 performance with 400 instances dataset

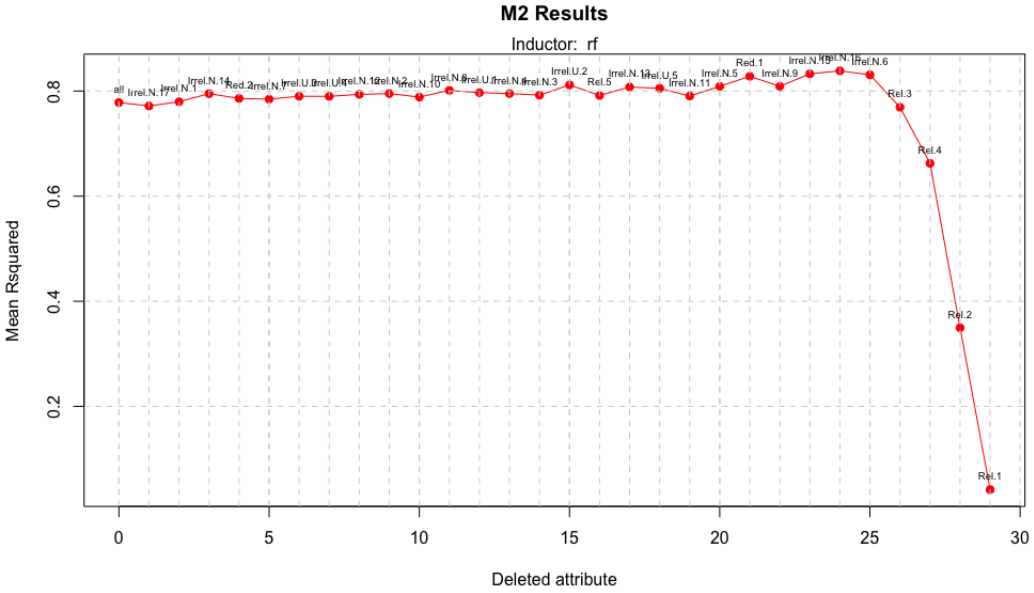

Figure 55: Model 2 performance with 800 instances dataset

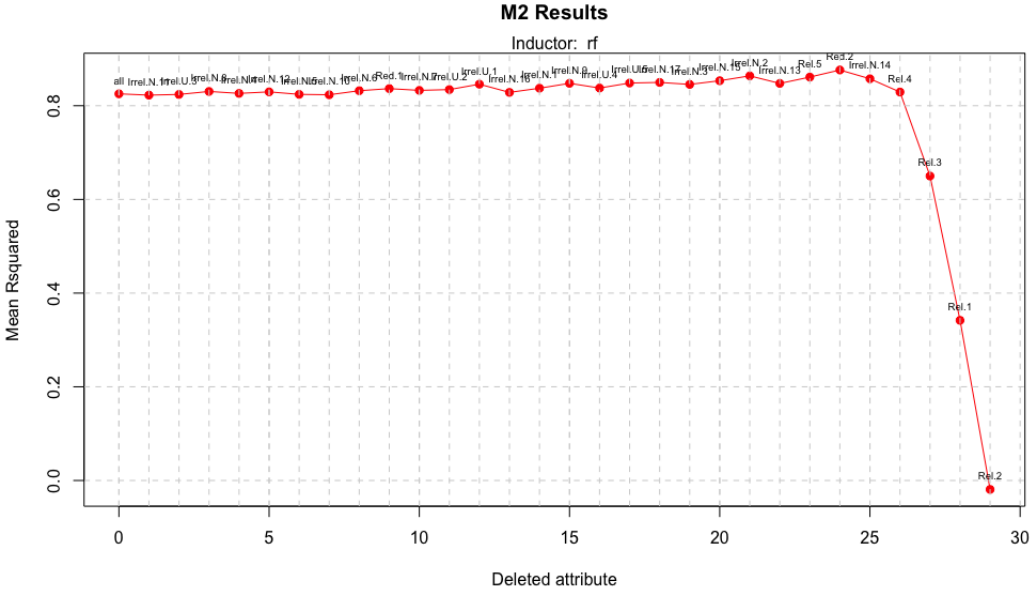

Figure 56: Model 2 performance with 1600 instances dataset

Appendix E Model 2 classification results

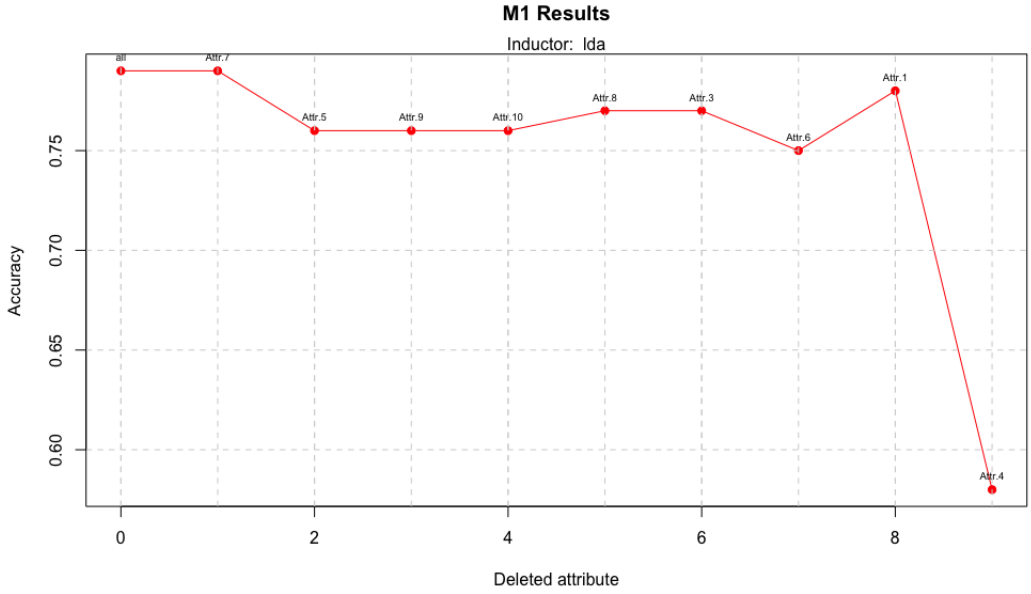

Figure 57: Model 2 performance with 100 instances dataset

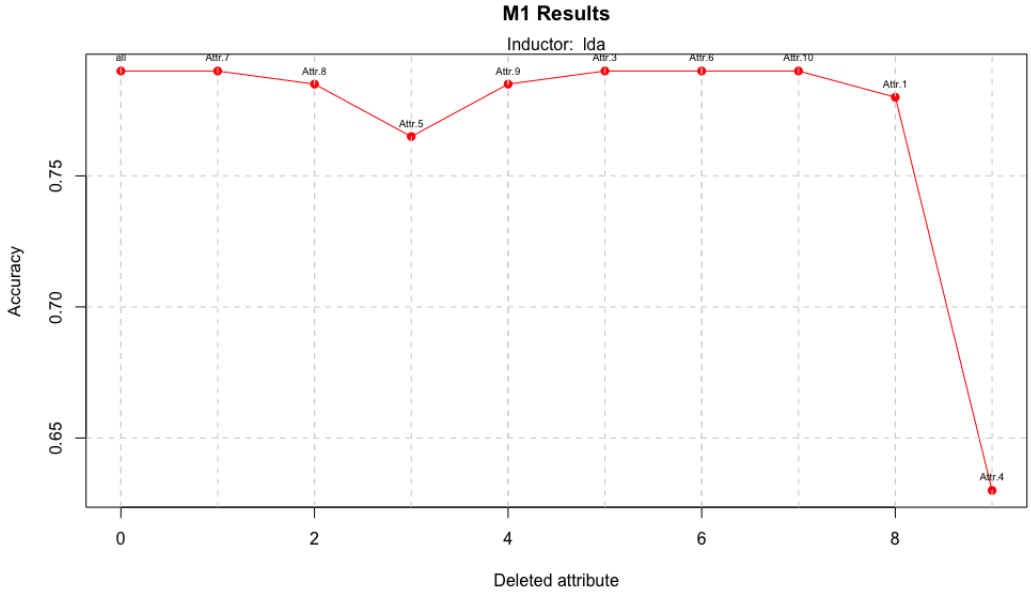

Figure 58: Model 2 performance with 200 instances dataset

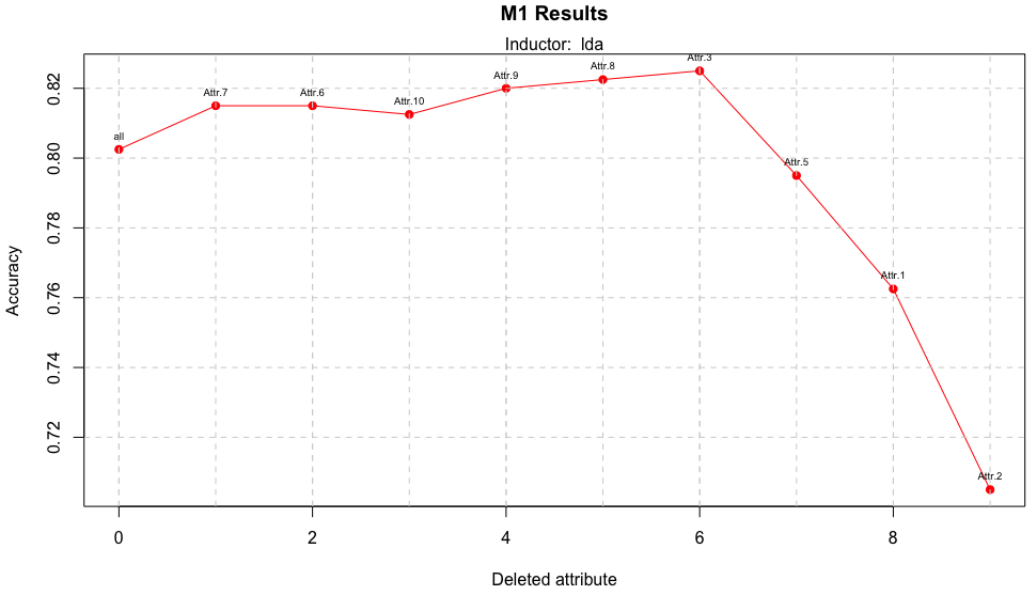

Figure 59: Model 2 performance with 400 instances dataset

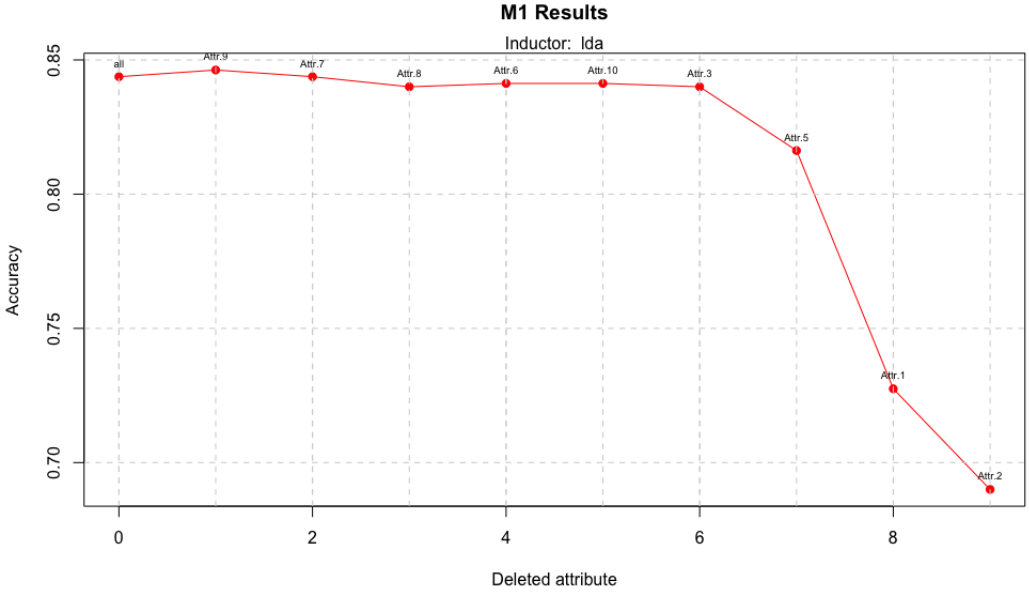

Figure 60: Model 2 performance with 800 instances dataset

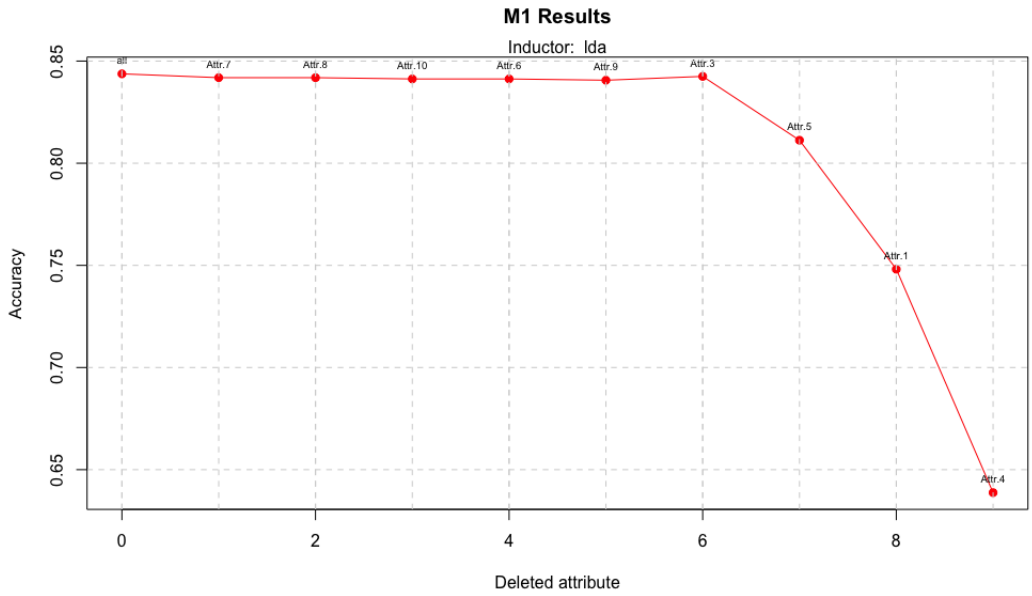

Figure 61: Model 2 performance with 1600 instances dataset

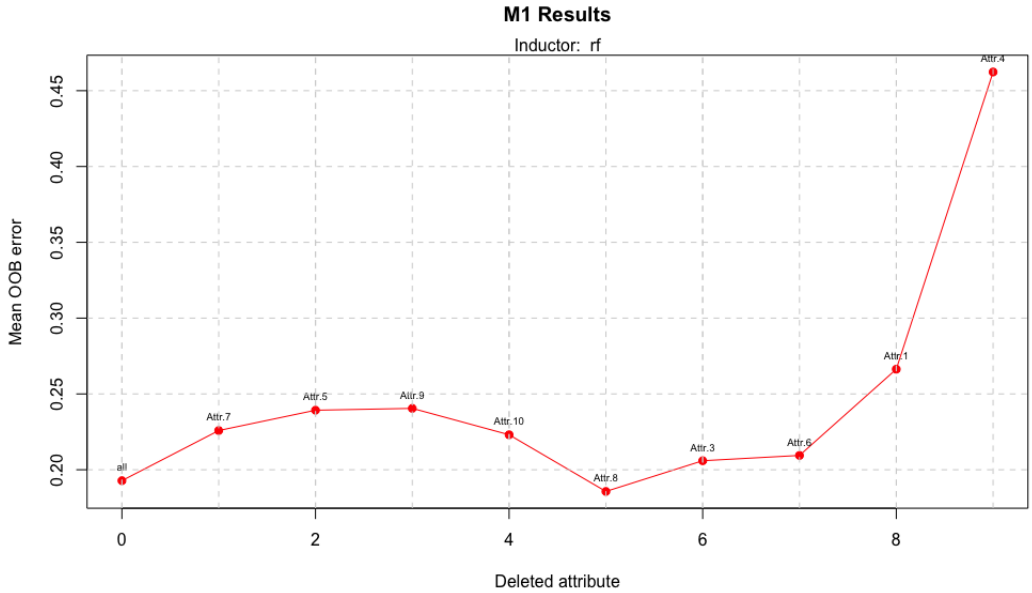

Figure 62: Model 2 performance with 100 instances dataset

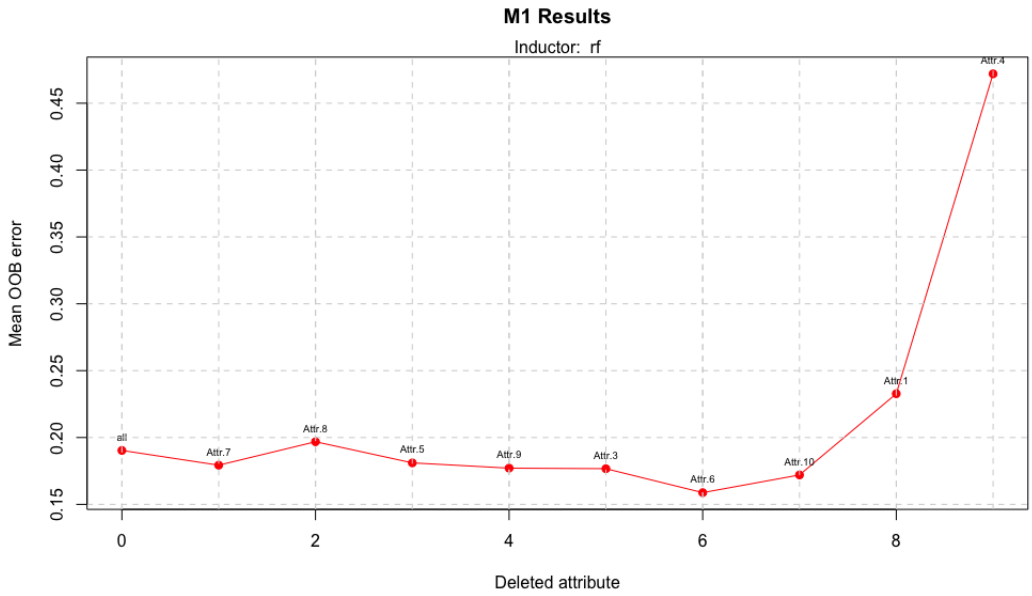

Figure 63: Model 2 performance with 200 instances dataset

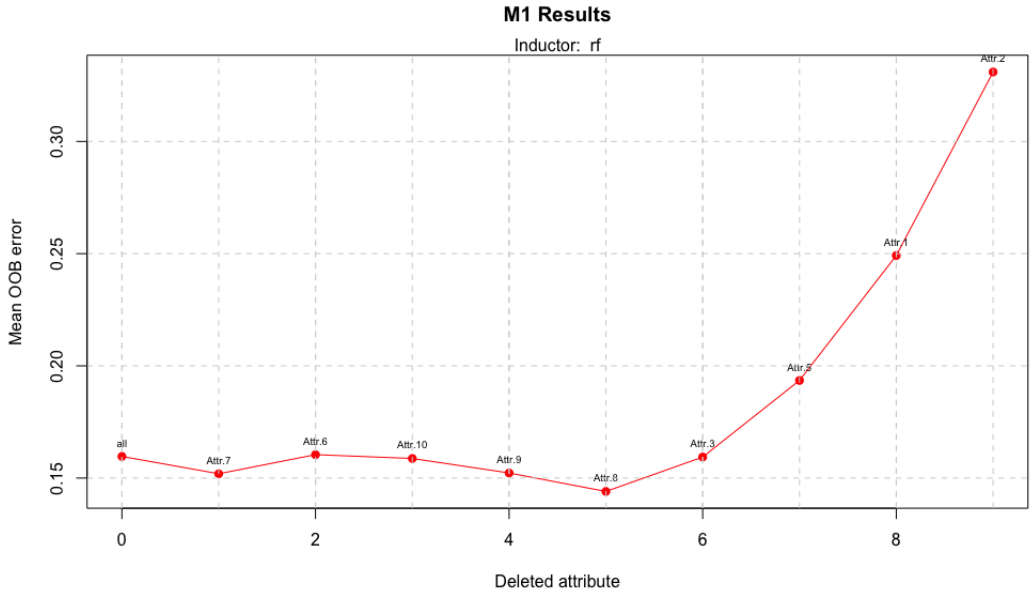

Figure 64: Model 2 performance with 400 instances dataset

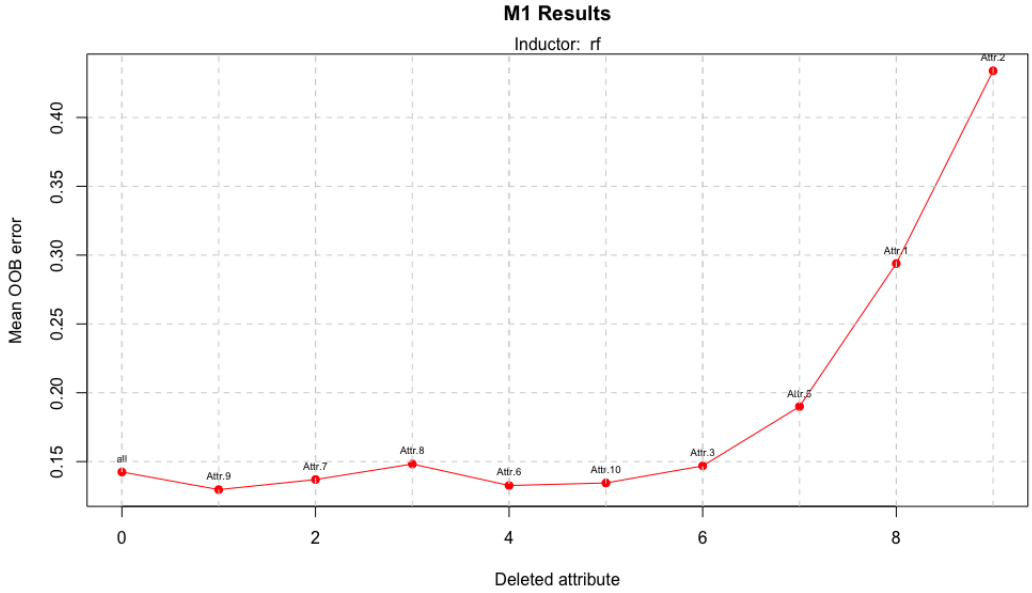

Figure 65: Model 2 performance with 800 instances dataset

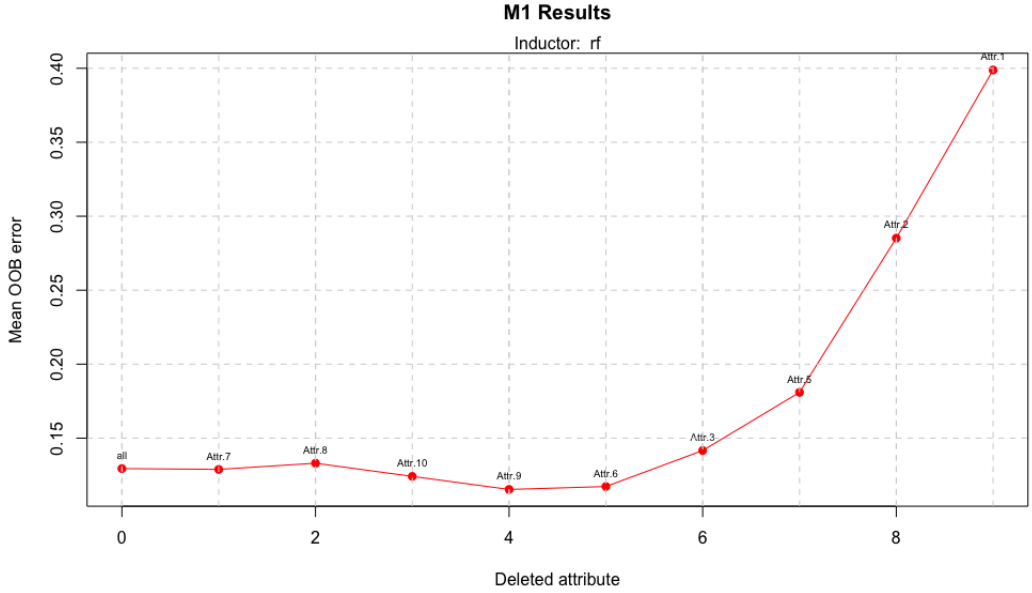

Figure 66: Model 2 performance with 1600 instances dataset